



VCU

Virginia Commonwealth University
VCU Scholars Compass

Theses and Dissertations


Graduate School

2022

SMART CITY MANAGEMENT USING MACHINE LEARNING TECHNIQUES

Mostafa Zaman

Follow this and additional works at: <https://scholarscompass.vcu.edu/etd>

 Part of the [Artificial Intelligence and Robotics Commons](#), [Controls and Control Theory Commons](#), [Other Electrical and Computer Engineering Commons](#), and the [Power and Energy Commons](#)

© The Author

Downloaded from

<https://scholarscompass.vcu.edu/etd/7170>

This Thesis is brought to you for free and open access by the Graduate School at VCU Scholars Compass. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of VCU Scholars Compass. For more information, please contact libcompass@vcu.edu.

©Mostafa Zaman, December 2022

All Rights Reserved.

SMART CITY MANAGEMENT USING MACHINE LEARNING TECHNIQUES

A Thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in Electrical & Computer Engineering in the Department of
Electrical & Computer Engineering at Virginia Commonwealth University.

by

MOSTAFA ZAMAN

Bachelor of Science, Bangladesh University of Engineering and Technology, 2015

Committee Chair : Dr. Sherif Abdelwahed,
Professor, Department of Electrical & Computer Engineering

Virginia Commonwealth University

Richmond, Virginia

December, 2022

Committee Members

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

Supervisor : Dr. Sherif Abdelwahed

Professor

Department of Electrical & Computer Engineering

Virginia Commonwealth University

Richmond, Virginia, USA.

Committee Member : Dr. Carl Elks

Associate Professor

Department of Electrical & Computer Engineering

Virginia Commonwealth University

Richmond, Virginia, USA.

Committee Member : Dr. Milos Manic

Professor

Department of Computer Science

Virginia Commonwealth University

Richmond, Virginia, USA.

Acknowledgements

Before anything else, I want to thank my adviser, Dr. Sherif Abdelwahed, for his help throughout my time in graduate school and beyond. I'll never be able to repay him for all the help he's given me over the past four years. Thanks to his insightful criticisms, enthusiastic support, patient mentoring, and extensive expertise, this dissertation would not exist. Foremost. His suggestions immensely helped me while researching and writing this thesis. There is no one I would have preferred to have as my adviser and mentor during my MSc. studies or continuation of my Ph.D. studies. A special thanks go out to Dr. Carl Elks and Dr. Milos Manic, two members of my committee, for their thoughtful critiques and recommendations, expert suggestions. My family members have been more instrumental in this process than anybody else. I'm grateful to my beloved mother and father, whose unconditional support and wise counsel have always been there for me. I also want to express my deepest gratitude to my loving wife, Nishat Salsabil, and everyone who has shown me support through their prayers and kindness. Finally, I would be remiss in not mentioning Sujay Saha, whose suggestions, support, and love helped me during the difficult times of my life.

Author's Declaration

I certify that the work presented in this thesis is my original work and has not been previously submitted for credit toward another degree. Almost all of the work in the lab was done by myself, and everyone who helped was given credit. Each piece of literature and resource used in support has been appropriately cited. I certify that I am the author of this thesis, that I have the right to use the material presented herein, that it has not been submitted for credit toward any other degree or professional qualification, and that all attributions in the text are accurate. My examiners have approved this authentic, final version of my thesis. I consent to my thesis being made publicly available in electronic format.

Dedication

I want to thank my parents, the late Parveen Sultana Zaman and Md. Mokhe-suzzaman, for the unconditional love they have shown me throughout my life and for helping me get to where I am now, which is why I am dedicating my Thesis to them.

TABLE OF CONTENTS

Chapter	Page
Examining Committee	i
Acknowledgements	ii
Author's Declaration	iii
Dedication	iv
Table of Contents	v
List of Tables	viii
List of Figures	ix
Abstract	xiii
1 Introduction	1
1.1 Motivation	3
1.2 Summary of Contributions	4
1.2.1 Smart Energy Aspect	5
1.2.2 Smart Transportation System Aspect	6
1.2.3 Smart Control in Water Distribution System Aspect	6
1.2.4 Smart Environment Aspect	7
1.3 Thesis Organization	8
2 Literature Reviews and Related Works	9
2.1 Smart Energy : Photovoltaic (PV) Generation	9
2.2 Intelligent Transportation System (ITS)	10
2.3 Smart Environment (AQI Evaluation)	12
2.4 Smart Water Distribution System (WDS)	14
3 Theoretical Background	16
3.1 Emergence of Artificial Intelligence (AI)	16
3.2 Introduction of Machine learning	17

3.2.1	Introduction of Deep Learning	20
3.2.1.1	Importance of Deep Learning	20
3.3	Convolutional Neural Network (CNN)	21
3.3.1	Benefits of Employing CNNs	22
3.3.2	General Architecture of Convolutional Neural Network	23
3.3.2.1	Convolutional Layer	23
3.3.2.2	Pooling Layer	25
3.3.2.3	Activation Function (non-linear)	26
3.3.2.4	Binary Step Function	27
3.3.2.5	Linear Activation Function	27
3.3.2.6	Non-Linear Activation Function	28
3.3.2.7	Sigmoid Activation Function	29
3.3.2.8	Tanh Function (Hyperbolic Tangent)	30
3.3.2.9	Rectified Linear Unit (ReLU) Function	31
3.3.2.10	Softmax Activation Function	32
3.3.2.11	Choosing the right Activation Function	32
3.3.2.12	Fully Connected Layer	33
3.3.2.13	Loss Functions	34
3.3.2.14	Introduction of Loss Function	35
3.3.2.15	Mean Absolute Error (MAE)	35
3.3.2.16	Mean Squared Error (MSE)	36
3.3.2.17	Root Mean Squared Error (MSE)	36
3.3.2.18	Huber Loss	37
3.3.2.19	Binary Cross-Entropy Loss	38
3.3.2.20	Categorical Cross Entropy Loss	38
3.3.2.21	Hinge Loss	38
3.3.2.22	Regularization to CNN	39
3.3.2.23	Optimizer Selection	41
3.3.2.24	Improving performance of CNN	43
3.4	Long Short Term Memory	43
3.4.1	General Architecture	43
3.4.2	Cell State	44
3.4.3	Deleting Information	45
3.4.4	Adding New Information	45
4	<i>OpenCity</i> Architecture	48
5	A Deep Learning Model for Forecasting Photovoltaic Energy with Uncertainties	50

5.1	Proposed Methodology	50
5.1.1	Dataset Description	52
6	Incorporation of Physiological Features in Drowsiness Detection Using Deep Neural Network Approach	54
6.1	Proposed Approach	54
6.1.1	Dataset Description	56
7	Air Quality Prediction using Distributed LSTM Approach for Smart City Testbed	59
7.1	Air Quality Index (AQI)	59
7.2	Methodology	60
7.2.1	Training	62
7.2.1.1	Forecasting	62
7.2.1.2	Classification	63
8	Adaptive Control for Smart Water Distribution Systems	64
8.1	Testbed Design	64
8.1.1	Instrumentation and Control Design	65
8.1.2	Safety System Design	66
8.1.3	Embedded System and Communication Architecture	66
8.2	Control System Design	67
8.2.1	Single-loop Feedback Control	67
8.2.2	Adaptive Feedback Control	68
8.2.3	Disturbances and User Consumption Modeling	68
9	SIMULATION RESULTS	71
9.1	Performance Evaluation	71
9.2	Smart Energy	73
9.3	Smart Transportation System	78
9.4	Smart Environment	79
9.5	Smart Water Distribution System	80
9.5.1	Discussion	83
10	Conclusions and Future Works	87
10.1	Smart Energy	87
10.2	Smart Transportation System	88
10.3	Smart Environment	88
10.4	Smart Water Distribution System	89

Appendix A Abbreviations	91
References	92
Vita	113

LIST OF TABLES

Table		Page
1	Selection of Appropriate Activation Function	33
2	Network Configuration for the proposed CNN-LSTM architecture	52
3	Network configuration for the proposed CNN architecture	58
4	Air Quality Index Categories [151]	60
5	Pollutant-Specific Sub-indices for the Air Quality Index (AQI)[154]	60
6	Network Configuration for the proposed LSTM architecture	62
7	User profile and time intervals, represented by the parameter vector θ	70
8	Confusion Matrix	72
9	Performance comparison of the proposed approach and the baseline	78
10	Results for the proposed methodology	79
11	Performance comparison of the proposed approach and the baseline	85
12	Performance comparison of the proposed approach and the baseline	85

LIST OF FIGURES

Figure	Page
1 An illustration of the position of deep learning (DL), comparing with machine learning (ML) and artificial intelligence (AI) [78, 79]	17
2 A map of Machine learning techniques [88]	19
3 Common Architecture of a Convolutional Neural Network [104]	22
4 Convolutional Layer	24
5 Max pooling Layer	25
6 Binary Step Function [113]	27
7 Linear Activation Function [113]	28
8 Sigmoid Activation Function [113]	29
9 Tanh Activation Function [113]	31
10 ReLU Activation Function [113]	32
11 Softmax Activation Function [113]	33
12 Fully Connected Layer	34
13 Regularization in Convolutional neural networks [120]	39
14 Long Short Term Unit (First Step)	44
15 Long Short Term Unit (Second Step)	45
16 Long Short Term Unit (Third Step)	46
17 Long Short Term Unit	47
18 The <i>OpenCity</i> platform architecture	49

19	Proposed CNN-LSTM framework	51
20	Proposed Algorithm for Drowsiness Detection	55
21	Proposed Architecture of Drowsiness Detection	56
22	Camera and Sensors Architecture	57
23	The Proposed Distributed LSTM Architecture	61
24	The Proposed flow chart of the Suggested method	61
25	A simplified process flow diagram for the water distribution system. Details for the business and hospital buildings are the same as the residential building and omitted to avoid cluttering the diagram.	64
26	MQTT (Message Queuing Telemetry Transport) communication for the testbed	66
27	Pressure control schemes for the water distribution system	67
28	A signal representation of user consumption	69
29	Prediction Comparison using CNN-LSTM over the whole learning period	73
30	Prediction Comparison using CNN-LSTM over the time sequence 9000- 10000	74
31	Prediction Comparison using CNN-LSTM over the time sequence 76000- 81000	74
32	Prediction Comparison using CNN-LSTM over the time sequence 99000- 100000	75
33	Training and testing data loss value using proposed method	75
34	Training and testing data loss values using baseline method	75
35	Prediction Comparison using LSTM over the whole learning period	76
36	Prediction Comparison using LSTM over the time sequence 9000-1000 . .	76

37	Prediction Comparison using LSTM over the time sequence 76000-81000 .	77
38	Prediction Comparison using LSTM over the time sequence 99000-10000 .	77
39	Training and testing data accuracy using the proposed method	78
40	Training and testing data loss values using the proposed method	78
41	Confusion matrix of the proposed approach	79
42	Forecasting Result for CO	80
43	Forecasting Result for NO ₂	80
44	Forecasting Result for Ozone	81
45	Forecasting Result for PM _{2.5}	81
46	Forecasting Result for PM ₁₀	82
47	Forecasting Result for SO ₂	82
48	Forecasting Result for Pb	83
49	Pressure profile for the two experimental control schemes vs different usage patterns. For the usage patterns, "S" is for the Same floor, and "O" is for the Opposite floor. The fixed pressure scheme utilizes one pressure setpoint for all usage patterns, while the dynamic pressure scheme varies the pressure set point according to the usage pattern.	84
50	Flow rate variation for the two pressure control schemes for a fixed user. For the dynamic pressure scheme, the flow rate for two different users at two different floors is shown. "U" stands for the Upper floor, and "L" for the Lower floor.	84
51	Power consumption per user for the two pressure control schemes.	86

Abstract

SMART CITY MANAGEMENT USING MACHINE LEARNING TECHNIQUES

By Mostafa Zaman

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical & Computer Engineering in the Department of Electrical & Computer Engineering at Virginia Commonwealth University.

Virginia Commonwealth University, 2022.

Director: Dr. Sherif Abdelwahed,
Professor, Department of Electrical & Computer Engineering

In response to the growing urban population, "smart cities" are designed to improve people's quality of life by implementing cutting-edge technologies. The concept of a "smart city" refers to an effort to enhance a city's residents' economic and environmental well-being via implementing a centralized management system. With the use of sensors and actuators, smart cities can collect massive amounts of data, which can improve people's quality of life and design cities' services. Although smart cities contain vast amounts of data, only a percentage is used due to the noise and variety of the data sources. Information and communication technology (ICT) and the Internet of Things (IoT) play a far more prominent role in developing smart cities when it comes to making choices, designing policies, and executing different methods. Smart city applications have made great strides thanks to recent advances in artificial intelligence (AI), especially machine learning (ML) and deep learning (DL). The applications of ML and DL have significantly increased the accuracy aspect of

decision-making in smart cities, especially in analyzing the captured data using IoT-based devices and sensors. Smart cities employ algorithms that use unlabeled and labeled data to manage resources and deliver individualized services effectively. It has instantaneous practical use in many crucial areas, including smart health, smart environment, smart transportation system, energy management, and smart water distribution system in a smart city. Hence, ML and DL have become hot research topics in AI techniques in recent years and are proving to be accurate optimization techniques in smart cities. In addition, artificial intelligence algorithms enable the processing massive datasets and identify patterns and characteristics that would otherwise go unnoticed. Despite these advantages, researchers' skepticism of AI's sometimes mysterious inner workings has prevented it from being widely used for smart cities. This thesis's primary intent is to explore the value of employing diverse AI and ML techniques in developing smart city-centric domains and investigate the efficacy of these proposed approaches in four different aspects of the smart city such as smart energy, smart transportation system, smart water distribution system and smart environment. In addition, we use these machine learning approaches to make a data analytics and visualization unit module for the smart city testbed. Internet-of-Things-based machine learning approaches in diverse aspects have repeatedly demonstrated greater accuracy, sensitivity, cost-effectiveness, and productivity, used in the built-in Virginia Commonwealth University's real-time testbed.

Keywords: Machine Learning, Artificial Neural Network, Convolutional Neural Network, Deep Learning, Smart City, Adaptive Control

CHAPTER 1

INTRODUCTION

Around 66% to 70% of the world's population will live in cities by 2050 [1, 2]. This rapid urbanization will have far-reaching consequences for city infrastructure, ecology, and safety. Several governments have advocated the notion of smart cities to optimize energy usage and manage the rapid rise in urbanization. Several countries (e.g., the United States, the European Union, Japan, etc.) have efficiently planned and implemented smart city initiatives to meet these potential future issues. Successful information and communication technologies (ICTs) [3, 4, 5] are necessary to manage the data analysis, data communications, and effective implementation of complex strategies required for a smart city's safe and efficient running [6].

Many smart city applications rely heavily on the Internet of Things (IoT) to produce massive data [7]. However, decisions based on extensive and intricate datasets are tough to predict. Artificial intelligence (AI), machine learning (ML), and deep reinforcement learning (DRL) are cutting-edge methods that may be used to analyze vast data and provide the best feasible option [6, 8, 2]. Furthermore, increasing the training data will boost the learning capacities of the methodologies above, leading to greater accuracy and precision. This results from the benefits of computerized decision-making [6, 9].

The success of the smart cities project relies on advancements in diverse areas such as intelligent transportation, cyber-security, smart grids (SGs), and 5G network architecture. In addition, the analytical and practical application of AI, ML, and DRL-based methodologies to improve the efficiency and scalability of a smart

city project are all heavily influenced by big data. Moreover, artificial intelligence's influence on our regular lives is growing. For example, artificial intelligence (AI) profoundly affects how people think about and interact with the world at work [6]. Many current plans for urban development suggest integrating cutting-edge operational technology into a city's fundamental infrastructures, including its transportation network, government services, and power grid [6].

The concept of "smart cities" is fast gaining traction as a viable option for the growth of metropolitan areas, intending to improve their residents' chances of survival and living. They astound us with their ingenuity in finding ways to use limited resources better, minimizing the adverse effects of our modern lifestyles on the natural world. Machine Learning allows computers to learn by making correlations between data. Moreover, by solving interconnected problems like optimizing urban planning and integrating city services for individualized results concerning the use of specific services by the inhabitants, machine learning can also encourage changes in utility business models, bringing greater mobility and comfort to users [10].

For Machine Learning (ML) to be effective, it must be able to generalize and resolve beyond the instances included in the training set. This is because it is highly far-fetched that these particular cases would be observed again in the predicted time frame, regardless of how much data is available for model training. As the predictions would be no better than a random guess if the model is too general, the model needs to be able to monitor input data and simulate the phenomena in a way that is reasonable and reasonable. On the other hand, too much transparency indicates that the model has been trained on adorned input data and cannot generalize to new data [10, 11, 12].

The motivation of this thesis is described in Section 1.1. Section. 1.2 provided the summary of contributions of this thesis. Finally, The thesis's architecture is laid

forth in section 1.3.

1.1 Motivation

Smart systems in healthcare, energy, transportation, water supply, power consumption, the environment, and other related networks are made possible due to the rapid development of AI, sensor technologies such as the Internet of Things (IoT), and wireless communication. Smart cities are a new type of integrated entity made possible by the interoperability of various smart systems and the gadgets used by their inhabitants. Over the past decade, communication technologies have enabled the interconnection of diverse heterogeneous networks, and autonomous agents that make up the Internet of Things (IoT) [13, 14]. High-performance sensors and end-user devices connected to the Internet of Things are the catalysts for using networks to make cities more intelligent and sustainable. Smart cities aim to provide integrated management systems with diverse, intelligent infrastructures to help traditional cities face future difficulties [15].

It might be difficult to make judgments in smart cities because of the many direct and indirect aspects and criteria involved. Our goal in writing this thesis was to zero down on one of the four main features of "smart cities," namely, efficient energy use, a well-developed public transportation system, and a healthy environment. It takes time and effort to make the best possible decision across the many facets of smart cities that use many devices and systems. Regularly collected data may help us make optimal choices. This means there is room for development in how decision support systems are trained [15, 16].

It's fascinating how major metropolitan areas gradually give up their urban chic in favor of a more technologically advanced identity. Technology, especially AI, is expanding the concept of "smart cities" (AI). More and more people are choosing

to make their homes in urban areas, which increases the demand for a unified infrastructure that can efficiently and affordably accommodate this burgeoning urban population. However, an increasing population poses difficulties for smart city planning and puts tremendous pressure on society to foster surroundings that are both innovative and sustainable. Thus, modern developed cities call for integrated smart policies and creative ways to improve monitoring functionality and promote urban living. For this reason, implementing different machine-learning approaches in distinct aspects of smart cities will help us to build a general machine-learning framework to monitor, organize, plan and improve the quality of services [15, 17].

In this thesis, we explore using four distinct machine-learning techniques in the context of smart city implementations. First, we use diverse ML strategies in smart power, transportation, and ecology. Then, we attempt to implement adaptive control in the smart water distribution system. Finally, to maximize the many facets of smart cities, we propose combining all the modules into a comprehensive machine-learning framework that includes data capture, analysis, and decision-making.

1.2 Summary of Contributions

The main contributions of this thesis are to implement different machine learning approaches to various aspects of smart city applications and investigate the usefulness of these approaches to increase the quality of the citizen's life. These approaches may be included as a module of a general machine-learning framework for the smart city. Furthermore, these modules can be a lower-level specification of a decision support system that will be utilized to handle uncertain aspects and events and tackle human intervention during emergencies. The following sections provide the contributions of each of the four elements of smart cities.

1.2.1 Smart Energy Aspect

This study suggests a hybrid CNN-LSTM model forecast PV output reliably. The presented model uses a convolutional layer (CNN) that functions as a buffer by extracting the data's local characteristics and then passing them on to the subsequent layers. The second layer (LSTM) extracts data from temporal and sequential features. Predictions of PV generation at one-step ahead and multiple-step-ahead horizons are made using the suggested approach [18] using a real dataset (DKASC, Alice Springs). A single LSTM and a BiLSTM prediction model are used to evaluate the proposed method's efficacy. Our framework's ability to learn temporal and local data aspects allows for accurate prediction performance. In addition, the model works well over a range of periods. The main contributions of this study include:

1. Presenting an uncertainty-aware deep learning algorithm for predicting PV generation profiles. Unlike prior studies in the field, which were typically developed for indirect forecasting of PV production (based on predictions of climatic variables), our proposed model predicts PV outputs by directly learning from raw data. Therefore, our model is robust against uncertain data (noisy, inaccurate, and missing data).
2. Compared to the previous PV output forecasting methodologies presented in the literature, our proposed algorithm uses a combination of CNN and LSTM; CNN extracts the local features from data, and LSTM captures the temporal relationships. In particular, the advantages of above two technologies are integrated to enhance the estimation accuracy and forecasting reliability.
3. The Performance of the proposed prediction model is evaluated on an actual dataset (DKASC), and the results are analyzed and compared to that of a single

LSTM and a BiLSTM neural network. Besides, the efficiency of our proposed model is validated for various prediction time horizons.

1.2.2 Smart Transportation System Aspect

This paper proposes a drowsiness detection method that integrates machine learning and physiological approaches such as heart rate and blood oxygen level. We have presented an efficient system to deal with real-time driver drowsiness detection using a convolutional neural network and other human biological features, including the blood oxygen level and cardiac rate.

The main contributions of this paper are:

1. Adding biological and physiological features in addition to neural network approaches to detect drowsiness,
2. Proposing a more accurate and robust detection method,
3. Integrating different neural networks and feature elements such as heart rate, blood oxygen level as a single input node of the neural network.

1.2.3 Smart Control in Water Distribution System Aspect

Realizing the significance of water and energy consumption and the opportunities offered by IoT technology, we took the initiative to build a water distribution system testbed as part of an extensive smart city project at Virginia Commonwealth University. This paper presents the design of the testbed and simulation results, comparing existing simple control approaches and an adaptive control approach that utilizes IoT data.

1. Illustrating how a water distribution system variables (pressure, flow, and en-

ergy consumption) are coupled and how they change with different usage profiles,

2. Showing that simple adaptive control approaches that utilize IoT data outperform existing classical control techniques for smart buildings,
3. Proposing more complex control configurations using the insight gained from simulation experiments.

1.2.4 Smart Environment Aspect

Despite the detrimental effects of air pollution, it is vital to have a reliable model for predicting AQI levels to improve urban public health and foster sustained social progress. Because of the direct influence on city administration and citizen health that air pollution forecasts have in densely populated places, such knowledge is of paramount relevance. Therefore, it is essential to use deep learning neural networks like long short-term memory (LSTM) to assist in policymaking that prioritizes improving air quality to create more sustainable cities. The suggested method may predict future AQI values by studying existing data on NO_2 , CO , O_3 , SO_2 , $\text{PM}_{2.5}$, and PM_{10} concentrations. Results from the current study show that the distributed LSTM technique outperforms standalone neural network predictors and regression models in predicting the AQI. the sample size indicates only a small time frame (one day). A more rigorous statistical analysis and more definitive conclusions might have been achieved with an extended period of hourly data.

In this paper, our primary contributions are presented below.

1. Introducing different distributed LSTM network approach for different pollutants to calculate AQI index values.

2. Utilizing distributed LSTM approach in a real-time constructed smart city testbed [19] to demonstrate a real-world use case.
3. Providing an in-depth analysis of different attributes for calculating AQI values for various pollutants based on sub-indices values.

1.3 Thesis Organization

The thesis is organized as follows. First, chapter 2 provides a detailed description of the Literature reviews and related works in all aspects. In chapter 3, all the theoretical aspects of the proposed methodologies are provided in detail. Third, chapter 4 provides a high-level description of our Open City Testbed. In chapter 5, A Deep learning model for forecasting photovoltaic energy with uncertainties is explained. Implying machine learning techniques in ITS and physiological elements are discussed in chapter 6. In chapter 7, Machine learning approaches for ensuring a smart environment of a smart city testbed are discussed. chapter 8 explains an adaptive control aspect of a smart water distribution system. chapter 9 provides the simulation results of implementing machine learning approaches in four domains of the smart city testbed. Finally, conclusions and future works are provided in chapter 10.

CHAPTER 2

LITERATURE REVIEWS AND RELATED WORKS

This chapter 2 provides the literature review for all four aspects of smart cities.

2.1 Smart Energy : Photovoltaic (PV) Generation

Authors in [20] proposed a short-term forecasting method based on support vector regression (SVR) models for PV output prediction. Their approach is then applied to forecast the power generation of a PV installed at VTech, USA. In literature [21], and [22], authors applied convolutional neural networks (CNN) to images/videos of sky and clouds, and based on CNN outputs, they analyzed the effect of clouds on PV generation. Authors in [23] presented a CNN-based prediction model for forecasting solar irradiance. They used solar irradiance predictions to estimate PV outputs.

In the studies mentioned above, climatic variables (such as solar irradiance, cloud position, and weather condition) are first predicted. Then, PV output predictions are generated based on the weather condition forecasts. However, for better accuracy, PV power generation is expected to be directly predicted from historical data samples (from previous PV generations and their related meteorological data). Therefore, deep learning approaches can be utilized to learn the features directly from the raw data and discover valuable representations.

Literature [24] and [25] used the long short-term memory (LSTM) approach to examine the impact of climatic conditions on estimating PV generation. Authors in [26] proposed a day-ahead PV power forecasting model by combining the deep learning modeling method with time correlation principles under a partial daily pat-

tern prediction (PDPP) framework. A hybrid deep learning technique for PV power forecasting is presented in [27]. Finally, the authors combined CNN with the LSTM approach for an end-to-end PV power prediction [28]. In [29], the authors applied a gated recurrent unit (GRU) neural network for generating hourly estimations of PV output.

The latest works demonstrate that deep neural networks outperform most PV prediction methods. Nevertheless, two empirical holes exist as unresolved questions in the literature. First, the estimation accuracy must still hit the realistic standard for real-world industrial applications. Second, most existing studies (including the above-mentioned deep learning methods) on PV output prediction only focus on one-step ahead forecasting; they do not offer flexible prediction horizons. PV generation can be predicted in a one-step or multiple-step horizon ahead. Short-term prediction horizons are usually chosen for estimating significant power ramp rates, while long-term forecasts help secure PV operations and enterprise participation. The accuracy of a PV power prediction model depends on the prediction horizon.

2.2 Intelligent Transportation System (ITS)

Machine Learning (ML) and computer vision allow computers to learn essential functions in the field of the smart transportation system. For example, the driver's sleepiness model should recognize whether the driver is tired, considering various situations and parameters. Based on deep learning techniques, researchers are constantly trying to develop an efficient, robust, and real-time driver drowsiness framework varying light intensities, size of images, and tasks regarding sleepiness [30, 31, 32, 33]. Based on changes in steering, speed, wheel movement, and pedal acceleration, this method attempts to gauge the driver's level of fatigue. Another option is to use a vision system that incorporates vehicle parameters, driving area, and lane-detecting

algorithms. Road conditions and driving ability are two of the many external elements affecting the outcome. This method uses the subjects' facial expressions to characterize tiredness. Based on the high association between drowsiness and the brain signal, a second strategy is to use physiological measurements [33].

On the other hand, a person's heart rate and oxygen level can give us the right and accurate idea of that person's physiological conditions. The pulse rate is the same regardless of heart stress and vibration in the arteries. Like other essential indicators such as blood pressure and breathing rate, the heartbeat varies with age and is a cause of cardiac depolarization. In adults, the average heartbeat of children is between 60 and 100 beats per minute [34, 35]. Approximately eighty beats per minute of the heart's rhythm are the ideal one while one achieves the desired state of relaxation [36]. Blood oxygen level is one of the key ingredients that keep human beings alive. Therefore, a real-time assessment of saturated human tissue blood oxygen (SpO_2) is crucial for tiredness. Many experiments indicate that SpO_2 in the conductor's body decreases as driving time and tiredness increase [37].

Autonomous systems that analyze driver fatigue and predict drowsiness may be essential for an intelligent vehicle to avoid critical accidents. The researchers used various machine learning and sensory procedures for detecting fatigue and exhaustion of the vehicle driver in recent days [31, 32, 33, 34, 35, 36, 37, 38]. Some approaches rely on tracking the driver's physiological characteristics such as cardiac rhythm, pulse rate, electroencephalography (EEG), electrooculography (EOG) signals [35, 38, 39]. Research has indicated that Alpha and Theta bands' EEG strength is increasing as the alertness level is declining [40], thereby offering signs of drowsiness. While some physiological signals have greater identification accuracy, they are only sometimes adopted because of their less practicality. The third set of strategies is focused on computer vision systems that can detect face changes during sleepiness [41, 42].

2.3 Smart Environment (AQI Evaluation)

Recently, there has been a lot of discussion on how artificial neural networks (ANNs) may be used to make accurate predictions about how much pollution will be in the air in the future. In addition, greater demand is being put toward improving urban air quality since it has been linked to the development of chronic diseases and the early deaths of people. Policymakers and metropolitan city planners aim to provide quick air pollution mitigation strategies that make minimal sacrifices in effectiveness [43]. Many applications for forecasting the near and far future have used ANNs in recent years with great success [44] [45] [46]. Additionally, an increasing number of doctors and artificial neural networks (ANNs) as a non-traditional data-driven method strategy based on hard science or determination [47, 48]. The use of ANNs, on the other hand, can be done without a thorough familiarity with the interplay between various factors and the concentration of air pollution. Finally, in recent years, powerful and less complex computational tools have been more widely available to the public to create and implement ANNs and their training algorithms [49].

While progress has been made in creating and understanding ANN models, their findings still need caveats despite the widespread success. The proliferation of neural networks and their application in generating massive amounts of data via the IoT has led to an explosion of research in many fields. There are three types of neural network models employed in these investigations: Multilayer Perceptron (MLP), Radial Basis Function (RBF), and Square Multilayer Perceptron (SMLP) for making predictions [50] [51]. In addition, traditional machine learning and neural networks demonstrate that various strategies incorporating meteorological and geographical factors may increase the precision of regional $PM_{2.5}$ predictions [52].

Environmental elements such as weather, the concentration of pollutants, the

closeness of receptors, and the area's geography all have a role in the accuracy of air quality forecasts. For example, the effects of the same contaminants on the environment change depending on the weather. On the same day, the concentration of pollutants affects air quality. The concentration of ambient air pollutants is most affected by meteorological conditions [53]. Traditional prediction approaches (such as mathematical statistics, multi-variable linear regression model, time series, and gray system) and machine learning methods (like artificial neural network (ANN) and support vector machine (SVM)) are now employed for air quality prediction [53].

Authors used linear and non-linear modeling to predict air quality [54]. In [55], the authors used SVM to predict air quality. At present, most studies use non-linear models to forecast air quality. A previous study in [56] showed that non-linear models (such as ANN) produce more accurate results than linear models because there are more apparent non-linear patterns in air quality data. ANN is a powerful tool for describing non-linear phenomena [57]. Therefore, ANN has been widely used in air quality prediction to preserve the integrity of the environment [53].

The authors employed an ANN model to forecast PM_{10} levels and discovered that pollution levels vary with seasons [58]. Second, Hourly $PM_{2.5}$ concentration in Santiago, Chile, was predicted using a feed-forward neural network developed by the researchers [59]. Third, to forecast and examine PM_{10} and $PM_{2.5}$ levels, authors employed a recurrent neural network (RNN) model. Fourth, by integrating RNN with data from natural sensors [44], Researchers enhanced their ability to forecast $PM_{2.5}$. Fifth, Pardo et al. merely employed a primary LSTM network to foretell air quality [60]. Finally, for $PM_{2.5}$ forecasting, a neural network is used for improvement with genetic algorithms [61]. In 2017, ozone concentrations over Seoul, South Korea, were predicted by Eslami et al. using a deep convolutional neural network (CNN) [62]. Finally, when developing their model for assessing air quality, Gu et al. relied on an

SVM model optimized with a hybrid of SAPSO and PSO [63].

2.4 Smart Water Distribution System (WDS)

The authors in [64] presented a water distribution testbed design (WADI). The primary purpose of the testbed is to research the structure of secure cyber-physical systems; hence the design architecture is not to facilitate system performance optimization. Furthermore, a Smart Water Distribution System (SWDS) architecture that uses IoT and ICT cloud computing technology is introduced in [65]. Finally, an IoT-based underwater water delivery grid architecture with water demand prediction is explained in [66].

The authors in [67] suggest a smart water distribution network solution for smart cities for the Indian scenario. An adaptive water demand forecast approach (WDF) is presented in [68] to assist smart water distribution systems in real-time operational management. A learning architecture for smart water distribution system control is developed in [69]. Finally, the authors in [70] combine artificial intelligence, ICT, and water conservation to solve water delivery problems and examine network optimization for drinking water distribution and wastewater storage networks.

The authors in [71] propose an intelligent water management system that utilizes IoT technology to coordinate business processes and support mechanisms for decision-making. The study highlights the critical advantages of using IoT data in water management. The work in [71] studies the use of IoT for water quality monitoring. The authors in [72] propose an IoT system for continuous monitoring in intermittent water distribution networks (IWDN). An IoT-Based Framework for Smart Water Supply Systems Management has been presented in [73] to avoid unwanted water accidents.

A control algorithm that considers the nonlinear relationship between pressure

and flows in distribution networks and utilizes discrete setpoints for pumps is described in [74]. The control methods for water distribution networks with the most frequent control goals, including duty pressure control, tank filling control, and energy output, are discussed in [75]. Modeling and regulation of a series of pumping stations providing water via pipes to intermediate storage tanks are discussed in [76], where the results are compared in real time with observed data. Finally, a flow control algorithm based on linearization to reduce leakage in water distribution networks is proposed in [77].

CHAPTER 3

THEORETICAL BACKGROUND

3.1 Emergence of Artificial Intelligence (AI)

This 3.1 talks about the connection among AI, ML and DL paradigms. Today, it's standard to use the phrases AI, ML, and DL interchangeably to refer to similarly intelligent-acting computer systems or programs. Figure 1 shows that DL is a subfield of ML and, by extension, artificial intelligence. Learning from data or experience automates analytical model construction, whereas AI often incorporates human behavior and intelligence into computers or systems [78, 79]. Deep learning (DL) also stands for data-driven learning techniques where computation is performed using multi-layer neural networks and processing. In creating a data-driven model, the term "Deep" in the deep learning approach refers to numerous layers or stages of processing data. Furthermore, DL approaches may play a significant role in advanced analytics and intelligent decision-making [80, 81], which is why the term "data science" is used to describe the complete process of discovering meaning or insights in data in a specific issue area [82].

Artificial intelligence encompasses methods that do not rely on "learning." For instance, symbolic artificial intelligence (AI) is concerned with hardcoding (i.e., explicitly defining) rules for every potential case in a particular domain of interest. Humans create these regulations based on their a priori understanding of the topic at hand and the desired outcome. While symbolic AI does well on tasks requiring low-level pattern recognition, such as text-to-speech or image-to-category translation, it struggles when faced with more complex tasks requiring high-level pattern

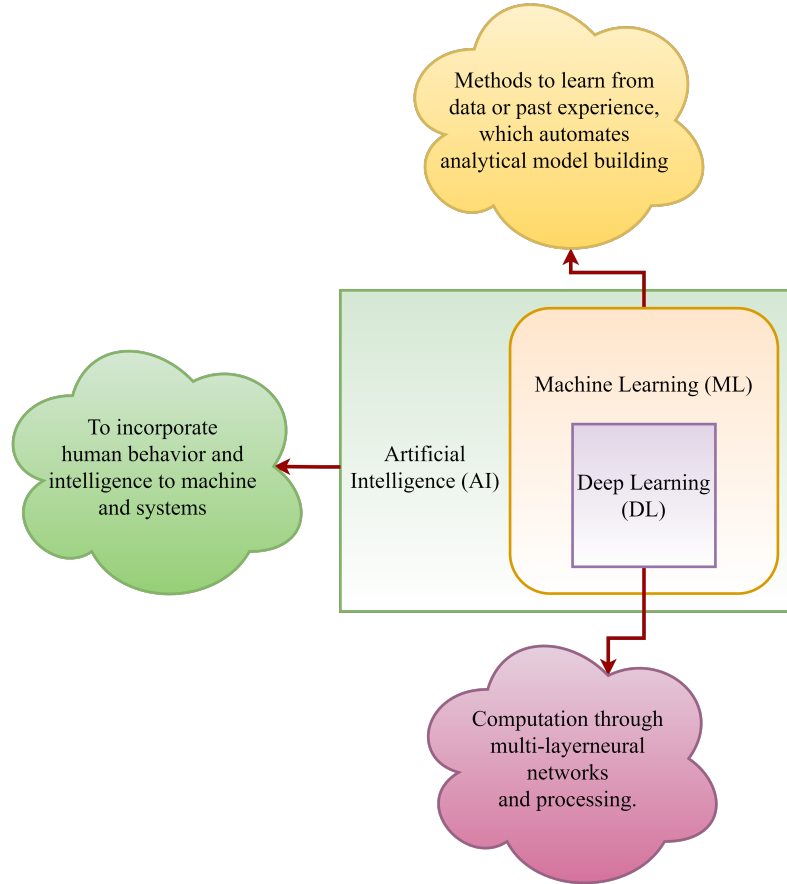


Fig. 1. An illustration of the position of deep learning (DL), comparing with machine learning (ML) and artificial intelligence (AI) [78, 79]

identification. ML and DL techniques excel in these more challenging endeavors [83].

3.2 Introduction of Machine learning

In many contexts, the two terms "machine learning" and "Artificial Intelligence" (AI) are used interchangeably or even confused with one another. The concept of artificial intelligence has expanded to incorporate areas formerly considered separate, such as computer vision, optimization, and data mining. One of the most critical capabilities in the evolution of AI is machine learning, a subfield or branch of AI that

seeks to solve complex problems by analyzing and learning from large amounts of data [84].

The term "machine learning" (ML) refers to a group of methods used to infer patterns in data and draw conclusions about the world. Algorithms are fed instead of hand-coded rules of "if" and "otherwise" decisions to analyze data [85]. These algorithms are built so that the program may learn from experience independently and with minimum human input. The computer is given the challenge of determining the best path from input to output, and it does so by repeatedly trying different options. Machine learning is a subfield of both computer science and mathematics. Probability theory, statistics, linear algebra, and calculus contribute to its theoretical foundation, which is organized mainly as layered algorithms. These methods rely on powerful computer hardware to analyze and discover patterns in massive datasets, such as determining the slightest error through gradient descent [84].

The multi-disciplinary field of machine learning has become ever more relevant and received massive attention as computers have become increasingly powerful over the years in companies with algorithmic advancements and the buildup of large databases. Algorithms are used to optimize hardware, and this interrelated development also enhances the collection and use of various data. Systems that only a few years ago performed significantly poorer than humans can now outperform humans on some specific tasks. For example, from a 72% accuracy in the image labeling contest, 'ImageNet' in 2010 to 96% accuracy five years later, the machine surpassed the human accuracy level of 95% in this specific contest [84, 86].

Many of these complex issues have generic ML algorithm solutions. These algorithms may be constructed without needing any specific planning ahead of time. However, an ML system must first train a model or set of rules using a labeled dataset to accurately predict the labels of data points (e.g., photos) not in the dataset [87].

All the machine learning techniques employed in different aspects are provided in Figure 2

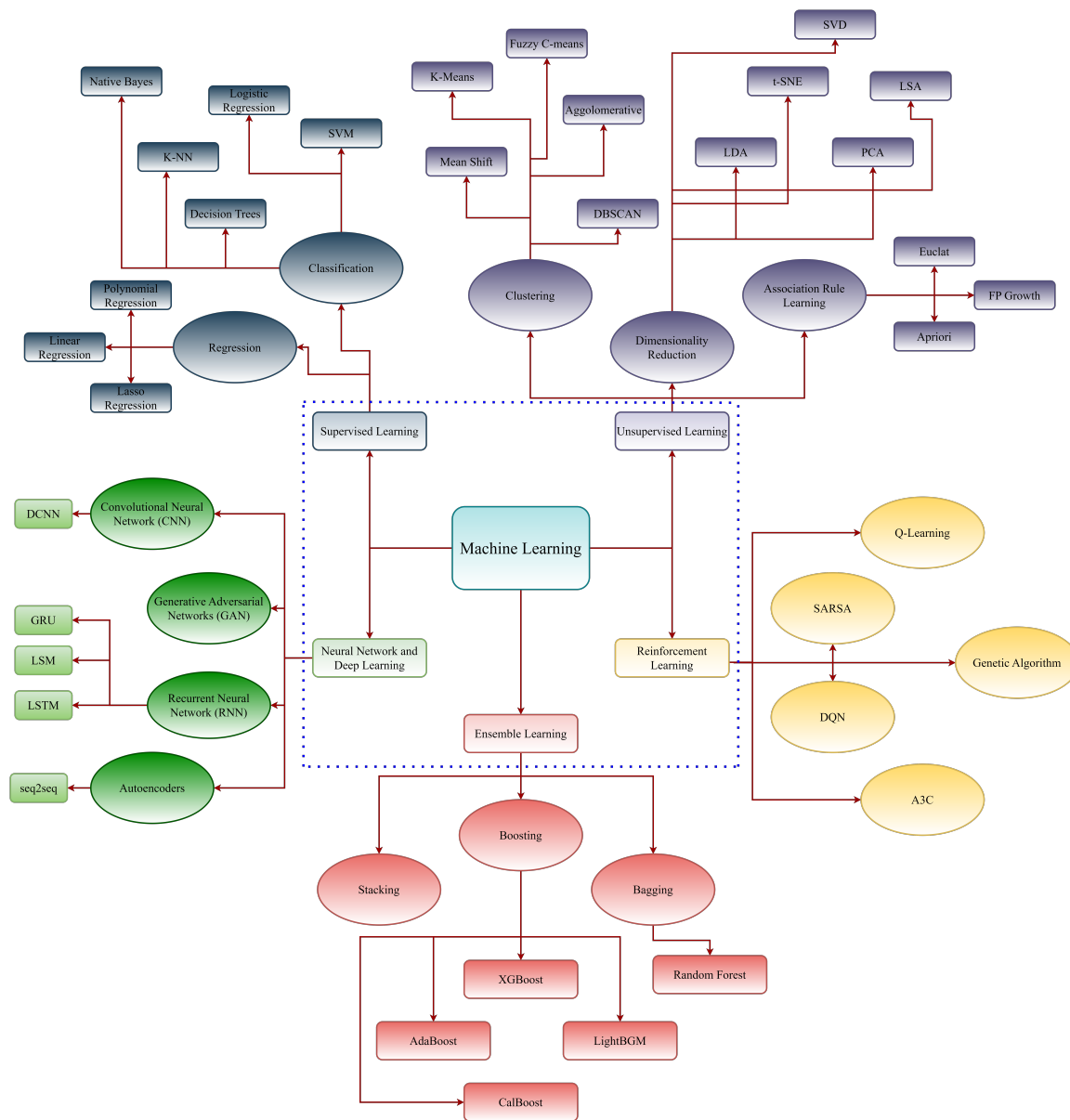


Fig. 2. A map of Machine learning techniques [88]

3.2.1 Introduction of Deep Learning

Deep learning aims to model data using elaborate structures that include several non-linear transformations. Combining neural networks into deeper networks is the fundamental building block of deep learning [84, 89]. Many different areas of study have significantly benefited from the recent developments in deep learning architectures, which have contributed considerably to the field of artificial intelligence [90, 91]. Face identification, audio recognition, computer vision, automatic language processing, and text categorization are just a few areas that have significantly benefited from these methods. Of course, different neural network architectures need clever stochastic optimization algorithms, initialization, and a clever structure choice. Nevertheless, they lead to awe-inspiring results, although only a few theoretical foundations are available till now [84, 91].

An area of machine learning called Deep Learning (DL) is influenced by how the human brain processes data. Contrarily, DL can function without humans having developed any rules; instead, it employs a significant quantity of data to map the input to labels. Layered algorithms (ANNs) are used in developing DL, each offering a unique interpretation of the input data [92, 89].

In contrast to traditional ML techniques, DL allows for the automated learning of feature sets across several jobs [89]. Due to the explosive development of the big data industry in recent years, DL has become a popular ML method [93, 30]. However, it is still a work in progress regarding innovative performance on various ML tasks [92].

3.2.1.1 Importance of Deep Learning

Where people cannot provide explanations for judgments based on their knowledge, deep learning can be helpful in cases such as image and object detection, medical

decisions, and speech recognition. Furthermore, it is necessary to use deep learning when the solution to a problem evolves, when adaptability to particular circumstances is required, or when the problem's scale is so enormous that it overwhelms our limited reasoning capacity, among other situations [92, 89, 91].

Several aspects of performance could account for deep learning's meteoric rise in popularity in recent years. First, due to its usefulness across various industries, DL is sometimes referred to as "universal learning." In DL methods, it is optional to have carefully developed features. The optimal characteristics are instead learned automatically in a contextually relevant way to the job at hand. The same DL method may be used across data types and even between applications, commonly referred to as transfer learning (TL). In addition, it's a helpful strategy for less data. Finally, scalability is not an issue while working with DL [94].

3.3 Convolutional Neural Network (CNN)

The versatility of CNNs has led to their widespread adoption in many areas, such as computer vision [95, 96, 97], audio processing [98, 99, 100], and face recognition [101, 102, 103], etc. CNNs, like traditional neural networks, are inspired by the structure of neurons in the brains of humans and other animals. Three significant advantages of CNN were outlined by [30]; they used similar representations, sparse interactions, and shared parameters. First, CNN uses shared weights and local connections to fully exploit 2D input-data structures like picture signals, as opposed to the traditional approach of using a single, global set of weights for the whole network. Fewer parameters are used in this procedure, making network training more straightforward and faster. Third, these functions are identical to cells in the visual cortex. These cells function as filters over the input, extracting the spatially-available local correlation. A typical form of CNN is structured similarly to the multi-layer

perceptron (MLP), with many convolution layers followed by sub-sampling (pooling) levels and finishing with FC layers. Figure 3 illustrates a typical CNN design.

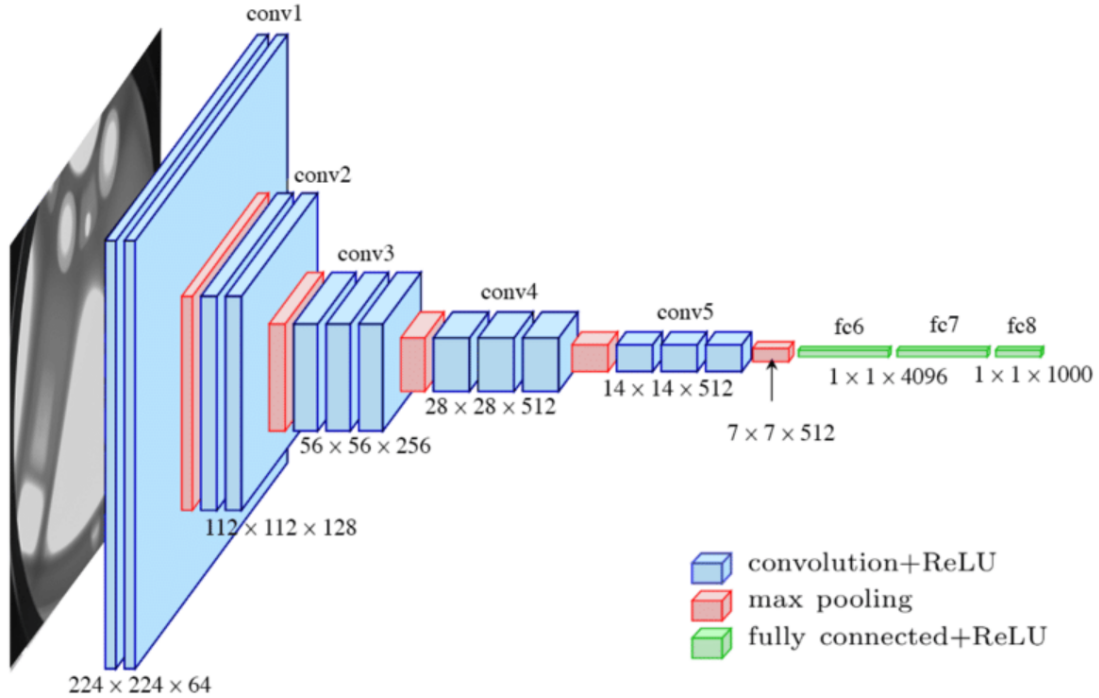


Fig. 3. Common Architecture of a Convolutional Neural Network [104]

3.3.1 Benefits of Employing CNNs

These are only some of how convolutional neural networks (CNNs) excel in computer vision above traditional neural networks, as described in this subsection 3.3.1. Convolutional neural networks (CNNs) have an essential characteristic called weight sharing [105], which allows them to be trained with fewer parameters, increasing generalization and decreasing overfitting. When feature extraction and classification [106, 107] are taught together, the resulting model output is organized and depends significantly on the generated features. If the input (often an image) has local spatial coherence, then the parameters of the network's convolutions can be shared, allowing

the network to utilize fewer weights. Also, by using a technique in the form of convolutions, they are ideally suited to extract helpful information at a low computational cost [108]. CNN is easier to implement in a large-scale network than other types of neural networks [92, 109].

3.3.2 General Architecture of Convolutional Neural Network

Convolutional neural networks (CNNs) are a deep learning technique to process data organized in a grid structure. CNN's are a form of deep learning method used to analyze data with a spatial or temporal component. They employ a succession of convolutional layers to increase their complexity, making them more challenging to implement than conventional neural networks. For Convolutional Neural Networks to function, convolutional layers must be present. subsection 3.3.2 discusses all the modules of a general CNN architecture.

3.3.2.1 Convolutional Layer

The convolutional layer is the primary building block in CNN design. A group of convolutional filters makes up the architecture. These filters are convolved with the input picture (represented as N-dimensional metrics) to produce a feature map. The kernel is defined as a grid of values or integers. The kernel weights are the individual numbers. At the outset of CNN training, random numbers are assigned to serve as the kernel's weights. The weights can be set in a variety of ways to add complexity. After each training round, the weights are fine-tuned, allowing the kernel to extract meaningful characteristics eventually [92, 109, 108].

The conventional neural network takes data in a vector format, whereas the CNN takes in data as a multi-channel picture. The grayscale picture format is an example of a three-channeled image, while the RGB format is single-channeled. Let's use a

4×4 grayscale picture and a 2×2 kernel with a random weight initialization to illustrate the convolutional procedure. The kernel initially moves horizontally and vertically over the whole image. Further, we calculate the dot product of the input picture and the kernel. Their values are multiplied and added in real-time to provide a single scalar. When additional sliding is no longer possible, the operation is repeated. The computed dot product values represent the feature map of the output. The two square kernels are shown in light green, while an input picture region of comparable size is shown in light blue. The product values are then multiplied by the sum of the input values, and the result is an entry value in the output feature map. However, in the preceding example, no padding was applied to the input picture [92, 109, 108].

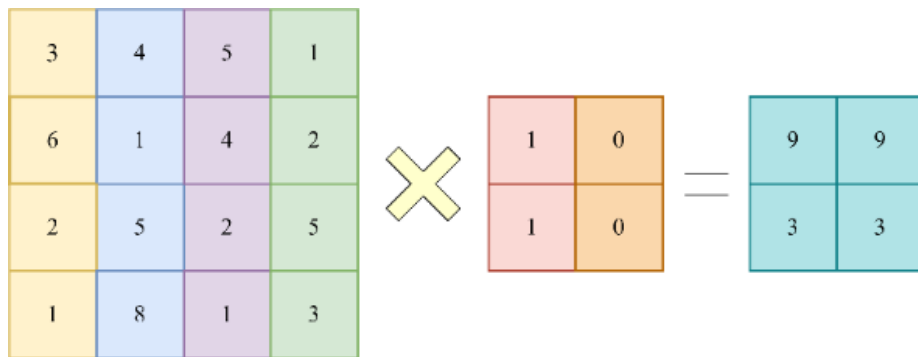


Fig. 4. Convolutional Layer

In contrast, the kernel is adjusted by applying a stride of 1 (where 1 represents the chosen step size's total vertical or horizontal positions). Padding is crucial in establishing input picture border size information as a counterpoint. In contrast, the characteristics on the border side are carried away at a breakneck pace. A larger input picture and feature map result from padding [92, 109, 108].

Figure 4 shows the convolutional layer in a CNN. In a neural network, each neuron in one layer is connected to every neuron in the layer below it. In contrast, CNN's have a limited number of weights accessible between layers. Since only a

limited number of weights or connections are needed, and since only a tiny amount of memory is required to store these weights, this strategy is memory-effective. Further, a matrix operation in CNN is significantly more expensive on the CPU than the dot (\cdot) operation [92, 109, 108].

3.3.2.2 Pooling Layer

Subsampling the feature maps is the primary responsibility of the pooling layer. The convolutional operations are followed to produce these maps. This method reduces the size of existing feature maps to generate more manageable intermediate maps. The majority of the most critical data (or characteristics) are preserved during the whole pooling process. In the same way that the stride and the kernel are given sizes before the convolutional procedure, the pooling operation does the same. Several different pooling techniques may be used in a variety of pooling configurations. These techniques range from tree pooling to gated pooling to average pooling to min/max pooling to global average pooling (GAP) to global max pooling. Maximum, minimum, and GAP pooling are the three most common and well-known approaches to this problem.

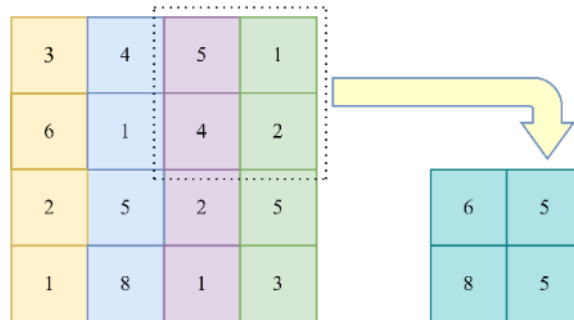


Fig. 5. Max pooling Layer

Figure 5 shows the maxpooling layer of a CNN. Since the pooling layer aids

the CNN in determining whether or not a given feature is accessible in the particular input picture but concentrates only on establishing the correct placement of that feature, it can sometimes lead to a drop in the overall CNN performance [92, 109, 108].

3.3.2.3 Activation Function (non-linear)

The primary purpose of all activation functions in all neural networks is to map the input to the output. The input value is calculated by adding the neuron's bias to the input data in a weighted fashion. In other words, the activation function generates the relevant output and uses it to determine whether or not to release a neuron based on the information. After all layers with weights (such as Fully connected and convolutional layers), a non-linear activation layer is used in CNN design. The mapping from input to output will be non-linear because of the activation layers' non-linear performance. The capabilities afforded by these layers allow CNN to learn more complex tasks. For example, error back-propagation may be used to train the network, but only if the activation function can discriminate, which is a crucial characteristic. Different activation functions are typically utilized for CNN and similar deep neural networks [92, 110, 111]. The Activation Function's main job is to convert the node's weighted sum of inputs into a value that may be used as output or passed on to the next hidden layer. An Activation Function's job is to take a node's input values and produce an output value (or a layer). An activation function is typically used to make a neural network less linear [92, 111]. Different activation function characteristics are provided in from 3.3.2.4 to 3.3.2.10

3.3.2.4 Binary Step Function

The threshold value determines whether or not a neuron is triggered in a binary step function. Neurons are engaged or deactivated depending on whether or not their input is more significant than a certain threshold, which prevents their output from being passed on to the next hidden layer. The use of binary step functions has certain restrictions. For example, it is unsuitable for issues requiring multiple values, such as multi-class classification, as it only produces single-valued results [112].

In mathematical notation, this looks like, and Figure 6 shows the graph of the binary step function:

$$f(x) = \begin{cases} 0, & a < 0 \\ 1, & a \geq 0 \end{cases} \quad (3.1)$$

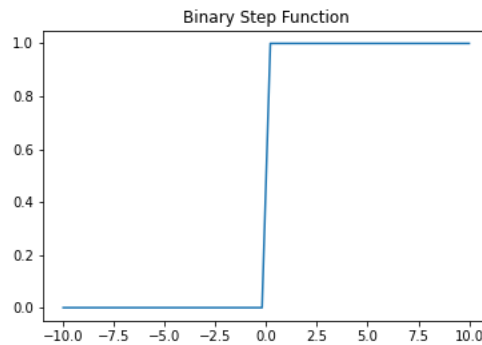


Fig. 6. Binary Step Function [113]

3.3.2.5 Linear Activation Function

When the activation is directly proportional to the input, we get a linear activation function or identity function [112].

$$f(x) = x \tag{3.2}$$

Figure 7 shows the graph of the Linear activation function. The function does not modify the input weights, only returning the original value. However, there are two fundamental drawbacks to using a linear activation function:

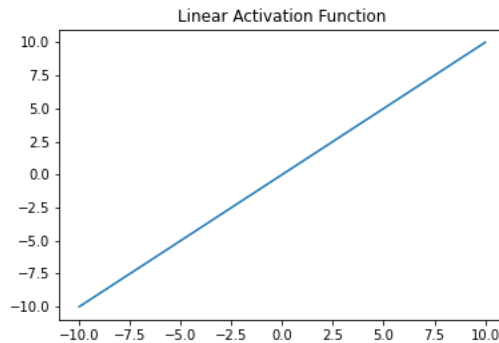


Fig. 7. Linear Activation Function [113]

When the derivative of a function is a constant and has no relationship to the input x , backpropagation cannot be used. If a linear activation function is used, all of the layers in a neural network will merge into a single one. This is because every neural network's output always has a linear relationship with its input at the first layer. Therefore, the neural network is reduced to a single layer using a linear activation function [112].

3.3.2.6 Non-Linear Activation Function

Because of its limited linear activation function processing capability, the model cannot generate very complex mappings between the network's inputs and outputs. These connections make backpropagation feasible since the derivative function can now be connected to the input. This allows a better understanding of which weights

in the input neurons may produce a more accurate prediction. Since the output would then be a non-linear mixture of input after passing through numerous layers of neurons, they make it possible to stack many layers of neurons. To a neural network, any output is just another functional calculation [92].

3.3.2.7 Sigmoid Activation Function

One of the most popular activation functions is the sigmoid activation function. This technique is frequently employed in modeling situations when a probability prediction is required. For this reason, the sigmoid distribution is optimal, as all probabilities lie between zero and one. This function's smooth gradient means its output values won't suddenly spike [112]. Figure 8 provides the graph of a sigmoid function.

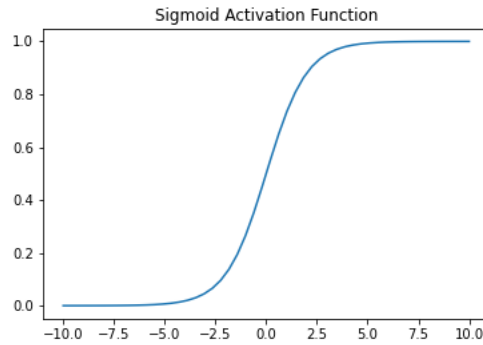


Fig. 8. Sigmoid Activation Function [113]

This function accepts a real number as input and returns a number between zero and one. As can be seen below, the closer the input is to a positive number (more significant), the closer the output is to a negative value (smaller), and the closer it is to zero, the opposite of one [112]. Mathematically it can be represented as:

$$f(x) = \frac{1}{(1 + e^{-x})} \quad (3.3)$$

The derivative of the function is

$$f'(x) = \text{sigmoid}(x) \times (1 - \text{sigmoid}(x)) \quad (3.4)$$

Networks that use sigmoid functions are susceptible to the Vanishing gradient problem, in which learning stops when the gradient value approaches zero. In other words, the logistic process does not produce symmetrical results near zero. So all the neurons will provide an output with the same sign. Consequently, this complicates and unstabilizes the neural network training process [112].

3.3.2.8 Tanh Function (Hyperbolic Tangent)

A Tanh function's output can be anywhere from -1 to 1, just like the sigmoid/logistic activation function, but the two are otherwise quite similar. So, Tanh operates on the principle that a positive input will provide an output value closer to 1.0, whereas a negative input will yield an output value closer to -1 [112].

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3.5)$$

Since the tanh activation function's output is centered at zero, we can simply map it to the extremes of the positive, negative, and neutral ranges. This function can be used typically in neural network hidden layers, whose values range from -1 to 0, producing a smooth median. It facilitates information concentration and encourages learning at a deeper level. It has the same issue as the sigmoid activation function with diminishing gradients. Also, unlike the sigmoid function, the tanh function has a steeper gradient [112]. Figure 9 provides the graph of a Tanh function.

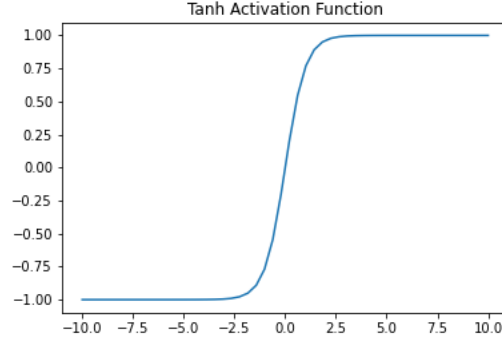


Fig. 9. Tanh Activation Function [113]

3.3.2.9 Rectified Linear Unit (ReLU) Function

ReLU is an abbreviation for "rectified linear unit." Despite appearing to be a linear function, ReLU plays a derivative role, supports backpropagation, and is computationally efficient. The key idea is that the ReLU function only triggers simultaneous activity in some neurons. However, if the linear transformation's output is less than 0, the neurons will be turned off [112]. Figure 10 provides the graph of a ReLU function. Mathematically it can be represented as:

$$f(x) = \max(0, x) \quad (3.6)$$

The ReLU function significantly outperforms the sigmoid and tanh functions in terms of computing efficiency because it activates just a small subset of neurons. In addition, because of its linear, non-saturating quality, ReLU hastens the convergence of gradient descent towards the global minimum of the loss function [112].

There is no gradient since the graph is negative. For this reason, weights and biases for some neurons are not changed during backpropagation. Due to this, some neurons may die since they are never stimulated. So, the model's capacity to effectively fit or train from the data is diminished since all the negative input values are

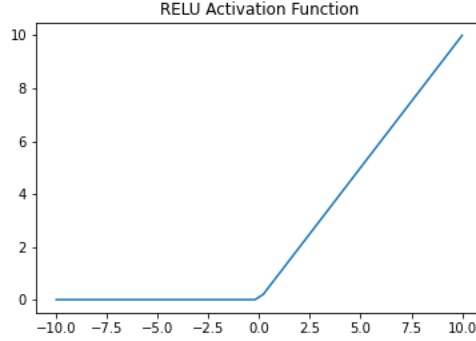


Fig. 10. ReLU Activation Function [113]

instantly converted to zero [112].

3.3.2.10 Softmax Activation Function

The sigmoid function gave results between 0 and 1, which may be interpreted as probabilities. In mathematical terms, the Softmax function may be considered a sum of many sigmoids. Essentially, it does a calculation of the odds of particular outcomes. The probabilities for each class are returned using the SoftMax function, which is analogous to the sigmoid/logistic activation function. Multi-class classification is often implemented as the activation function for the last layer of a neural network [112]. Figure 11 presents the graph of a softmax function. Mathematically it can be represented as:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \quad (3.7)$$

3.3.2.11 Choosing the right Activation Function

For each problem, activation functions must be fine-tuned based on the projected variable type [92]. Finally, here are a few rules for determining the activation function

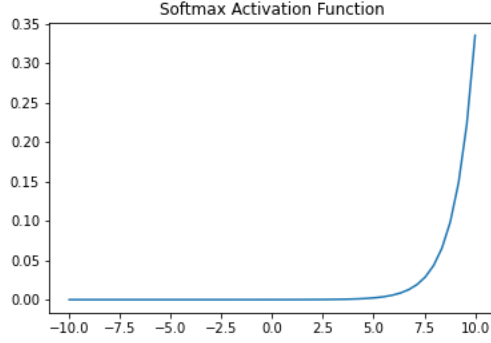


Fig. 11. Softmax Activation Function [113]

for the output layer based on the type of prediction problem provided in Table 1

Table 1. Selection of Appropriate Activation Function

Problem Specification	Types of Activation Function
Regression	Sigmoid/Logistic
Binary Classification	Sigmoid
Multiclass Classification	Softmax
Multilabel Classification	Sigmoid
Convolutional Neural Network (CNN)	ReLU
Recurrent Neural Network (RNN)	Tanh and/or Sigmoid

3.3.2.12 Fully Connected Layer

This layer is often the last in a convolutional neural network’s structure. In this layer, every neuron communicates with every other neuron in the layer below it. CNN uses it as its classifier. As a feed-forward artificial neural network, it is based on the same fundamental principle as the more commonplace multilayer perceptron. The FC layer takes information from the layer before the last convolutional pooling layer. When the feature maps are fattened, they are used to construct a vector that serves

as the input. According to Figure 12, the final CNN output is the data produced by the FC layer [84, 92].

CNN has a fully-connected architecture, meaning that each neuron in a layer works with all of the pixels in the input matrix, and no weights are shared between neurons in adjacent layers. Since learning new weights for each neuron is not essential, training time and costs may be drastically reduced by learning a single set of weights for the whole input [84, 92].

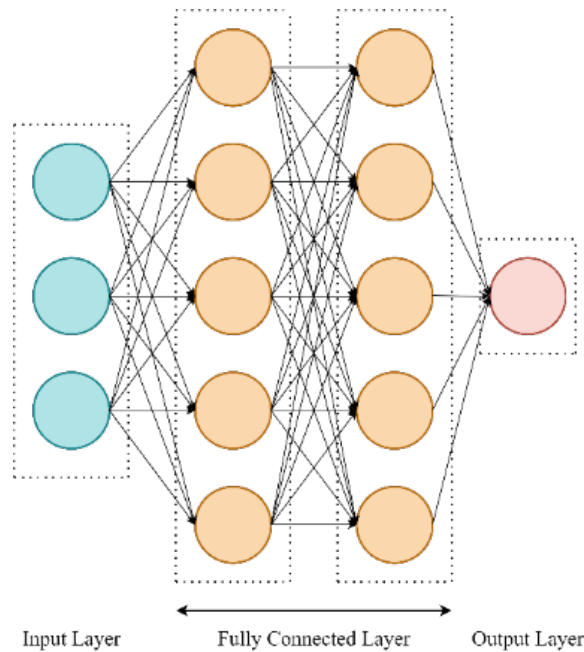


Fig. 12. Fully Connected Layer

3.3.2.13 Loss Functions

The final classification is achieved from the output layer, representing the CNN architecture's last layer. Finally, some loss functions are utilized in the output layer to calculate the predicted error created across the training samples in the CNN model. This error reveals the difference between the actual output and the predicted one.

Next, it will be optimized through the CNN learning process [92].

3.3.2.14 Introduction of Loss Function

The loss function provides information about the network's overall performance; a more considerable loss function indicates a less effective network. There are two main loss functions: classification loss and Regression loss. An example of classification loss is predicting an outcome from a set of category inputs. However, Regression Loss is employed when the task at hand involves regression, such as when trying to forecast continuous values [92]. The most widely used loss functions in Neural networks are explained in the following subsection 3.3.2.15 to - subsection 3.3.2.15 .

1. Mean Absolute Error (MAE)
2. Mean Squared Error (MSE)
3. Root Mean Absolute Error (RMSE)
4. Huber Loss
5. Cross-Entropy Loss
6. Hinge Loss

3.3.2.15 Mean Absolute Error (MAE)

Mean Absolute Error or L1 loss, is a loss function commonly used for regression issues. Calculated by taking the absolute value of each data point's deviation from the mean, it shows the gap between the raw data and the computed results. When given a set of cases with identical input feature values, MAE will find the median target value to be the most accurate predictor. If we look at Mean Squared Error,

where the average represents the best forecast, we can see how this measure stacks up. The magnitude of the gradient in MAE depends only on the sign of $y - \hat{y}$; thus, it will be considerable even when the error is minimal, which might cause issues with convergence [114, 115, 116]. We need to use mean absolute error in regression if we don't want outliers to skew your results. The knowledge that your distribution is multimodal and that it is preferable to have forecasts at one of the modes rather than at the mean can also be helpful [114, 115, 116]. The mathematical representation is given below Equation 3.8)

$$MAE = \frac{1}{n} \sum_1^n |y - \hat{y}| \quad (3.8)$$

3.3.2.16 Mean Squared Error (MSE)

A standard regression loss function is the Mean Squared Error (MSE)(given in Equation 3.9), often called L2 Loss. Specifically, the average difference between the original and anticipated values is the square root. Given several samples with the same input feature values, the ideal prediction is the mean target value due to MSE's sensitivity to outliers. By contrast, the best Mean Absolute Error forecast is the midpoint. Since MSE penalizes outliers more severely, it is appropriate to apply it if we expect our target data to follow a normal distribution given the input. Here, the MSE loss function can be employed [114, 115, 116].

$$MSE = \frac{1}{n} \sum_1^n (y - \hat{y})^2 \quad (3.9)$$

3.3.2.17 Root Mean Squared Error (MSE)

The root mean square error is a popular statistic used to evaluate the accuracy of a forecast. Using Euclidean distance, it displays how far the predicted value deviates

from the measured value. The root-mean-square error (RMSE) is calculated by first finding the residual (difference between prediction and truth) for each data point, then adding the residual norm for each issue, then finding the mean of residuals, and finally taking the square root of that mean. Since RMSE employs and requires exact measurements at each projected data point, it finds widespread usage in supervised learning applications. In addition, it is helpful to have a single metric in machine learning to evaluate a model's efficacy throughout all phases of its calculation (training, cross-validation, and monitoring after deployment) [117]. RMSE metric equation is expressed in Equation 3.10, which measures the standard deviation of prediction errors [114].

$$RMSE = \sqrt{\frac{1}{n} \sum_1^n (y - \hat{y})^2} \quad (3.10)$$

3.3.2.18 Huber Loss

Typically, Huber Loss is applied to regression issues. Since it only considers the error to be squared within an interval, it is less affected by extreme values than the MSE. Because it evaluates both the MSE and the MAE, the Huber Loss provides a good compromise. Huber Loss is effectively the best of both L1 and L2 losses, combining their respective strengths. For massive errors, it is linear, and for tiny mistakes, it is quadratic [115, 116]. Huber loss equation is expressed in (3.11). We can define this loss as follows:

$$L_\delta = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & \text{if } |(y - \hat{y})| < \delta \\ \delta((y - \hat{y}) - \frac{1}{2}\delta) & \text{otherwise} \end{cases} \quad (3.11)$$

3.3.2.19 Binary Cross-Entropy Loss

This loss function is typically applied when dealing with multigroup (two) classification. Cross-entropy quantifies the degree of dissimilarity between two random variables, whereas entropy characterizes the unpredictability of the processed information. Increased cross-entropy loss indicates a more significant discrepancy between the expected probability and the actual label [115, 116]. The loss equation is provided in (3.12) where y is the actual label and p is the forecasted value post hypothesis.

$$J = -(y \log(p) + (1 - y) \log(1 - p)) \quad (3.12)$$

3.3.2.20 Categorical Cross Entropy Loss

When applied to more than two categories, Binary Cross Entropy Loss transforms into Categorical Cross Entropy Loss. Therefore, the labels should be one-hot encoded when the categorical cross entropy loss function is employed. The vector will have precisely one non-zero component in this approach, as all the other components will be multiplied by zero [115, 116].

3.3.2.21 Hinge Loss

Hinge loss (provided in (3.13)) is another widespread option for loss functions used in classification. Hinge loss was first created with SVM to determine the maximum distance that may be maintained between the hyperplane and the classes. False predictions are punished by loss functions, whereas correct ones are not. Therefore, the target label's score should be (at least) one point higher than the total of all the wrong labels [115, 116].

$$SVM\text{Loss} = \max(0, 1 - y \cdot \hat{y}) \quad (3.13)$$

3.3.2.22 Regularization to CNN

The main problem with getting good generalizations from CNN models is overfitting. In the second situation (Figure 13), the model is considered overfitted since it performed extraordinarily well on training data but failed on test data (unseen data). On the other hand, if the model needs to pick up more information from the training data, it's under-fitted. Finally, when the model performs satisfactorily on the training and testing data, it is "just fitted" [92, 118, 119, 120].

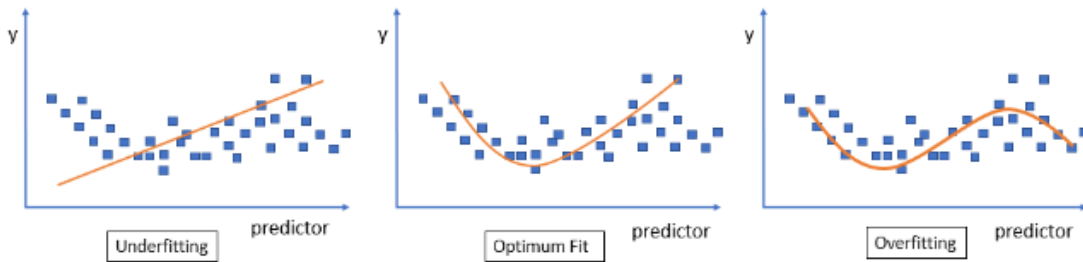


Fig. 13. Regularization in Convolutional neural networks [120]

1. Dropout: As a method of extrapolation, it is commonly employed. There is a random pruning of neurons throughout each training iteration. This pruning forces the model to learn a variety of independent characteristics, and it does so by apportioning the feature selection power over the whole group of neurons. During training, the lost neuron will not participate in back- or forward propagation. In contrast, predictions are carried out by the full-scale network as it is tested [92, 119, 120].

2. Drop-Weights: This approach is quite similar to the dropout method. The

main distinction between drop-weights and dropout is that in the former, the weights (connections) between neurons are removed rather than the neurons themselves [92, 119, 120].

3. Data Augmentation: A simple method to prevent over-fitting is to train the model using a large sample of data. Data augmentation is needed for this purpose. The size of the training dataset is artificially increased using several methods [92, 119, 120].

4. Batch Normalization: Batch normalization can be considered a pre-processing activity at each network layer. It's also used to lessen "internal covariance shift" in the activation layers. The internal covariance shift is defined by the differences in activation distribution between layers. This variation gets exceptionally high owing to constant weight updates throughout training. As a result, the model's convergence time and training time will rise. The CNN design has a layer for the batch normalization process to address this problem [92, 119, 120, 121]. Using batch normalization has these benefits:

1. It avoids the issue of the diminishing gradient. Second, it can mitigate the effects of a weak weight setup.
2. It speeds up network convergence dramatically (for large-scale datasets, this will be extremely useful).
3. It has difficulty reducing the training reliance between hyper-parameters.
4. It has little effect on regularization; therefore, the likelihood of over-fitting is diminished [92, 119, 120, 121].

3.3.2.23 Optimizer Selection

The goal of all supervised learning algorithms is to minimize the error. Loss functions are based on many different tunable parameters (such as biases, weights, etc.). Gradient-based learning methods are the go-to for building a CNN network. As many training iterations as possible, the network parameters should be updated continuously. The network should seek the locally optimal solution during training to reduce the error. Specifically, the amount of updates to the parameters is what is used to determine the learning rate. The training epoch is a single iteration of the parameter update over the whole training dataset. Even if it is a hyper-parameter, it is essential to pick the learning rate carefully so that it does not have an insufficient effect on the learning process [92, 122].

Gradient Descent algorithm: Every training period is a new opportunity for any algorithm to tweak the network's settings and get it closer to optimal performance. To properly update the parameters, the objective function gradient (slope) must be computed using the first-order derivative of the network parameters. The next step in minimizing the error is to update the parameter counterclockwise for the gradient. Finally, network back-propagation updates the parameters, with the gradient at each neuron. This process may be expressed mathematically as 3.14.

$$W_{ij}(t) = W_{ij}(t) - \Delta W_{ij}(t); \Delta W_{ij}(t) = \eta \frac{\partial E}{\partial W_{ij}} \quad (3.14)$$

The final weight in the training epoch is denoted by $W_{ij}(t)$, while the weight in the preceding (t-1) training epoch is denoted $W_{ij}(t)$. The learning rate is η , and the prediction error is E [92].

Batch Gradient Descent: This method's [123] implementation requires only a single lag update to the network settings across all training datasets. The parameters

are updated based on the calculated gradient of the entire training set. BGD helps the CNN model converge quickly for a short dataset while producing a more stable gradient. However, it takes time and effort because the settings are only adjusted once every training cycle [92].

Stochastic Gradient Descent: This approach [124] revises the settings with each new training sample. Every training epoch should begin with a selected subset of the training data. In addition to being quicker than BGD, this method also requires significantly less memory for large training datasets. However, the frequent updates make very noisy steps toward the solution, leading to volatile convergence behavior [92].

Mini-batch Gradient Descent: Mini-batch Gradient Descent is a method in which the training data are broken into several smaller batches. Each mini-batch is essentially a tiny, nonoverlapping sample set [125]. When a new mini-batch of data has been generated, the next step is to update the parameters based on the latest data. Combining the benefits of BGD with SGD gives this approach its edge. This algorithm converges reliably, uses less memory, and performs more computations per second [92].

Adaptive Moment Estimation (Adam): It's just another popular method of optimization. Recent developments in deep learning optimization are exemplified by Adam [126]. The Hessian matrix, which uses a second-order derivative, symbolizes this. Adam is a specialized learning method for developing deep neural networks. Adam's benefits include higher memory efficiency and lower computing requirements. Adam works by determining the adaptive LR value for each model parameter. It combines the best features of Momentum and RMSprop. Then, analogous to momentum, it takes a moving average of the gradient and uses the square of the gradient to scale the learning rate [127].

3.3.2.24 Improving performance of CNN

According to various deep learning applications [128, 129, 130], the most popular ways to boost CNN performance using data augmentation techniques or transfer learning for extensive data set, Expanding the training period, Changing the model architecture, adding regularization, and fine-tuning hyperparameters.[131].

3.4 Long Short Term Memory

3.4 provides the description of a Long short term memory units. In [132], Authors presented the artificial neural network (ANN) known as Long Short-Term Memory (LSTM). As time has passed, it has risen to prominence as a leading RNN. It is frequently used for issues involving sequential steps, such as writing recognition [133, 134, 135], Load forecasting [136, 137, 138], traffic forecasting [139, 140, 141], etc. In a recent study, researchers at Google propose the LSTM as a powerful and flexible framework for a range of machine learning applications. As a result of its resistance to explosions and, more specifically, the vanishing gradient problem, this solution has gained a lot of attention [92].

3.4.1 General Architecture

The LSTM's architecture is explained in 3.4.1. The LSTM architecture is unfurled in the diagram below. The leftmost cell represents the state of the LSTM at the previous time step, and the rightmost cell represents the state of the LSTM at the next time step. The present instant of time is roughly in the middle of the range. There are three incoming wires. First, the input X_t and the initial timestep's output are read at the bottom left corner, known as hidden state h_{t-1} . The next step is to join the input X_t with the secret state h_{t-1} and then send that combined signal via the

four gates shown in the diagram's yellow boxes. Finally, a straight arrow represents the third input from the primary cell, which enters the cell at its top. As a result of using this cell state, the LSTM can recall long-term dependencies with a far lower risk of disappearing and expanding gradient issues that plague conventional RNNs [84, 142, 143].

3.4.2 Cell State

The cell state, which is mathematically nothing more than a vector, can be envisioned as a superhighway for information that links all the nodes in a chain. It's a vital part of the LSTM since it helps the network remember past input dependencies over extended periods. Furthermore, it can read from, write to, and delete data from this internal storage. This additional information is added to the current state of the cell rather than multiplied by it, which is the key to avoiding the vanishing gradient problem. In addition, make sure gradients are distributed equitably, and that backpropagation doesn't use the chain rule. The LSTM cell has four distinct layers of a neural network, each performing a specific task. The three-layer sigmoid function is internally nested, and its output matrices have values between one and zero [84, 144].

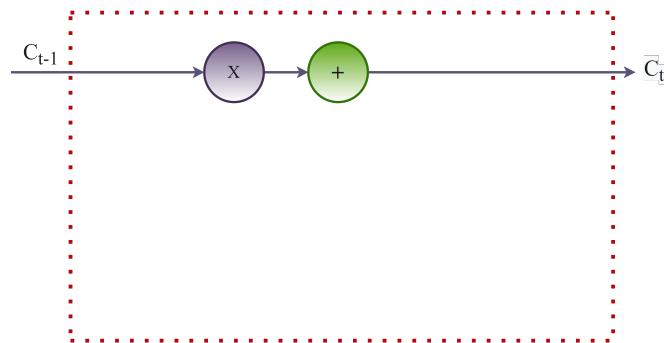


Fig. 14. Long Short Term Unit (First Step)

Figure 14 shows the first step of the LSTM operation. All the equations used in a LSTM operation are provided from Equation 3.15 to Equation 3.19.

3.4.3 Deleting Information

Forget gate assumes the current inputs X_t and the prior output h_{t-1} . When a neural layer's output is a matrix, the sigmoid function compresses the product of the current input and the weights into a matrix with values between one and zero. The forget gate is used to multiply by an essential factor, the cell state of the primary cell. The forget gate may be considered a reduction filter applied on the initial cell state to remove or downgrade values [84, 144]. Figure 15 shows the second step of the LSTM operation.

$$f(t) = \sigma(W_i \times [h_{t-1}, x_t] + b_f) \quad (3.15)$$

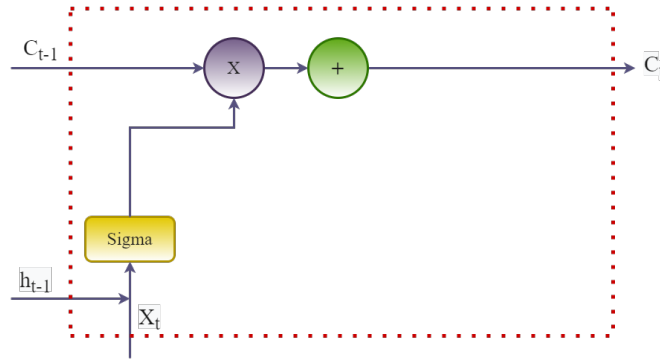


Fig. 15. Long Short Term Unit (Second Step)

3.4.4 Adding New Information

The sigmoid and tanh gates, the first and second input gates, are the following building blocks. The input gate filters the tanh layer in the same way as the forget gate. In addition, it determines how much data it will store from zero to one.

Figure 16 and Figure 17 provides the third and final step of the LSTM operation.

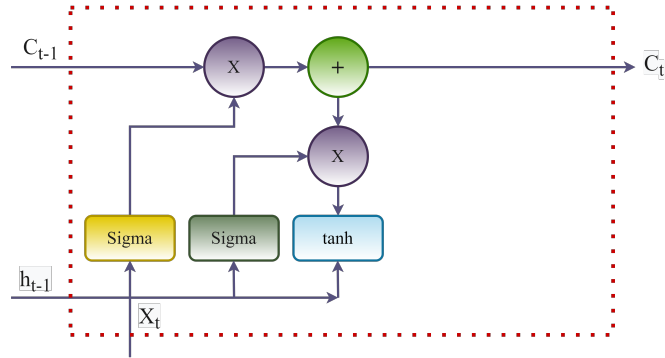


Fig. 16. Long Short Term Unit (Third Step)

$$i_t = \sigma(W_i \times [h_{t-1}, x_t] + b_{if}) \quad (3.16)$$

In contrast, the tanh activation function of the opposite gate generates potential new cell state values. The hyperbolic tangent, which lies between -1 and 1, is used. There appears to be some addition and subtraction of data when new candidate values are introduced to the cell state. The equation for the new values C'_t is provided below [84, 144].

$$C'_t = \tanh(W_i \times [h_{t-1}, x_t] + b_c) \quad (3.17)$$

$$C_t^i = C'_t \times i_t \quad (3.18)$$

Our computations have calculated many neural networks using the same inputs. We have used a sigmoid filter to select which memory data to store and which to discard (the initial state of the cell). Then we may update the cell state Ct by combining the previously updated form via the forget gate, element by element, with the newly proposed values C_t^i [84, 144].

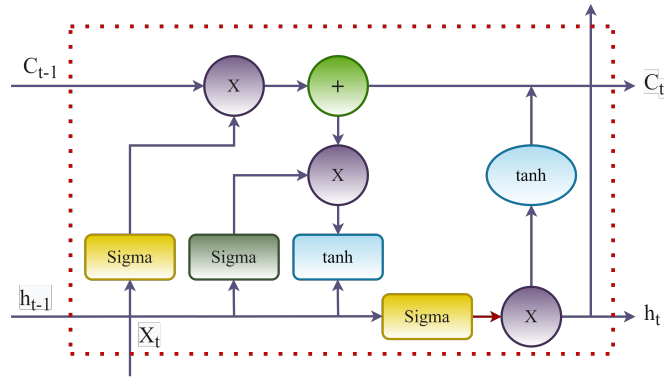


Fig. 17. Long Short Term Unit

In total the modifications to the cell state is:

$$C_t = f_t \times C_{t-1} + i_t \times C'_t \quad (3.19)$$

The last step is to determine the cell's output. The currently hidden state inputs are fed into a regular neural layer. This output is preceded by a sigmoid activation, which is pointwise multiplied by the squashed cell state in a Tanh layer to provide a scalar value between -1 and 1. Here, the output is filtered based on the cellular state. Therefore, the most crucial factors are the previous and present inputs. However, the cell state may alter the outcome by multiplying it with either positive or negative values [84, 144].

CHAPTER 4

OPENCITY ARCHITECTURE

This new testbed tackles the issues of plug-and-play, scalability, and interoperability through IoT-based communication technologies, autonomous systems, software development, and database administration. The team working on this project will also create a suite of distributed management control algorithms to aid in making smart city management decisions. Researchers will be able to work together and exchange data on the proposed platform, and a community of smart city aficionados will arise, all of whom will help to develop cutting-edge smart city apps [19].

Our work on the *OpenCity* platform is primarily concerned with diverse smart city applications, houses and buildings, transportation systems, water distribution systems, and the environment. Researchers will use data to create a management framework that can run any advanced infrastructure fully or partially independently. The suggested management structure would teach researchers to factor in various human elements. This testbed facilitates collaboration between government and industry partners to better comprehend analytics and smart technologies and quickly translate academic findings into practical applications. This smart city paradigm presupposes that all entities have some intelligent node that either conducts local sensing and control or uses control signals sent by edge or cloud management systems. [19].

The goal of this testbed is to provide a smart city experimental environment that includes [19]

1. Data collection, processing, and analytics,
2. An IoT communication network,

3. Real-time data visualization,
4. A Configurable and extendable management system

An example of a "block" in *OpenCity*, seen in Fig. 18, consists of four buildings arranged in a two-by-two grid and bordered by roadways. The area is centered on a four-way crossroads with many sets of traffic lights. It's possible to build a city grid out of these interlocking modules. The 'textitOpenCity' neighborhood has four structures: a) homes, b) offices, c) a healthcare facility, and d) a water distribution system. Automated cars, road sensors, and traffic lights will all be part of the transportation network, and they may share data with the textitOpenCity system [19].



Fig. 18. The *OpenCity* platform architecture

CHAPTER 5

A DEEP LEARNING MODEL FOR FORECASTING PHOTOVOLTAIC ENERGY WITH UNCERTAINTIES

5.1 Proposed Methodology

Figure 19 shows the proposed diagram of our proposed hybrid CNN-LSTM framework. We implemented the hybrid network for the PV power prediction problem. The CNN feature extraction functionality comprises a single 2D layer of convolution. The convolutional operation adds many convolutional layers sequentially, enabling the initial layers to learn features of low levels in the implemented input. The feature map that is the output of the convolutional layers has a boundary, which shows the input characteristics' exact position. This feature map implies that small moves at the input position would result in a separate characteristic map. A pooling layer is generally introduced to mitigate the feature map's invariance limit after the CNN layer. In contrast, the activated function improves the models' capacity to learn complex structures. Our proposed model passed the input data through a CNN layer that extracts the features for each time step.

We took 300 recent observations as a sample. Therefore, the output from the CNN layer had 300 different characteristics. This result was input into a Stacked LSTM model, which predicted future values. In our sequence learning module, we deployed many LSTM layers, each with n neurons. The network's whole hidden state sequence is generated by calculating the return sequence for the two LSTM layers; the last LSTM layer is fake, revealing the hidden state at the conclusion. Before the fully linked layer, we utilized the drop-out layer to avert overrun. As a result, the fully

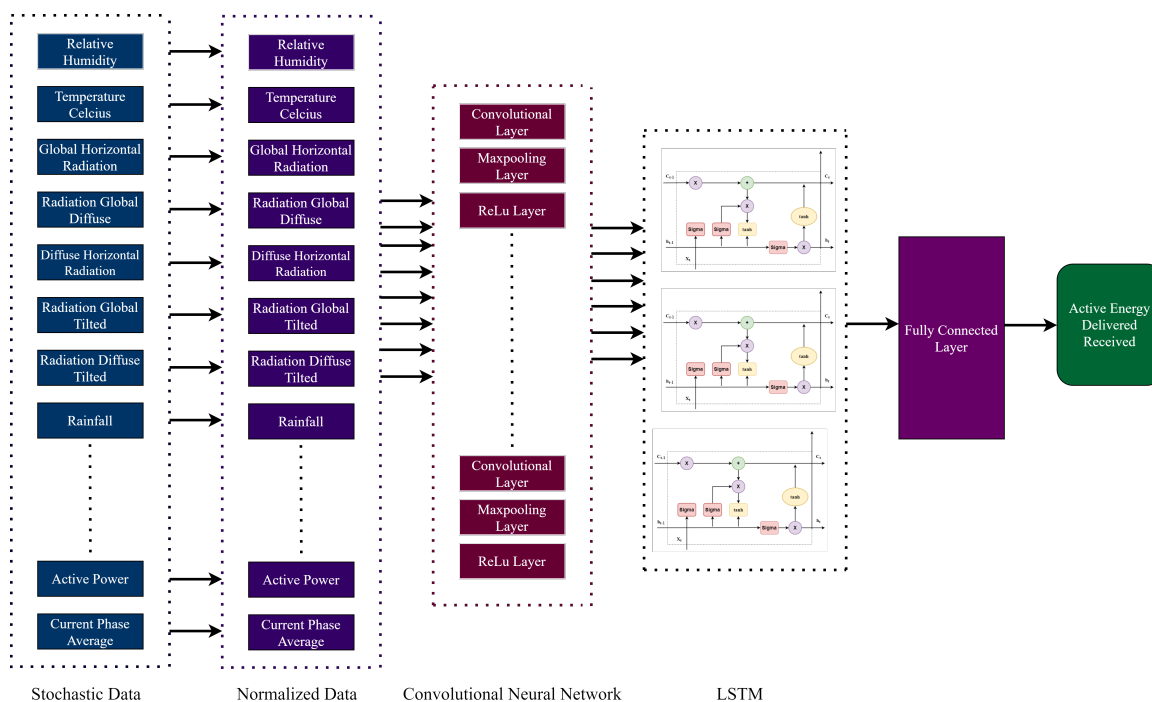


Fig. 19. Proposed CNN-LSTM framework

connected layer has n total neurons. In our experiment, we manipulate the number of neurons in the output layer to gauge how far into the future the network can project [145]. Our suggested hybrid CNN-LSTM model’s network design is detailed in Table 2.

The dropout layer is a great tool to solve the overlapping issue when developing a deep learning model. We placed a dropout layer between the CNN extraction block and the LSTM sequence to stop overfitting. Dropout and fully integrated layers are connected to the sequence learning block output to produce the results. To facilitate evaluation, we split the original convolutional layer into two and added a MaxPooling and Dropout layer. We combined a single CNN layer with a stacked LSTM model to improve the prediction performance.

Table 2. Network Configuration for the proposed CNN-LSTM architecture

Layer	Output Shape	Parameter
conv2d_3 (Conv2D)	(None, 300, 1, 1)	16
flatten_1 (Flatten)	(None, 300)	0
reshape_1 (Reshape)	(None, 300, 1)	0
lstm_3 (LSTM)	(None, 300, 50)	10400
lstm_4 (LSTM)	(None, 300, 50)	20200
lstm_5 (LSTM)	(None, 50)	20200
dense_1 (Dense)	(None, 1)	51

5.1.1 Dataset Description

DKASC, an Alice Springs dataset from Australia’s Northern Territory, was used for this research [18]. DKASC is a solar technology demonstration facility that has been operational for over ten years in the dry climate of Alice Springs, central Australia. We used ”combined output of all arrays” information from March 1, 2018, through February 28, 2019, and meteorological records from the same time frame. PV power generation is susceptible to solar irradiance levels. Additional environmental factors can affect PV power estimates, such as the surrounding temperature, the module’s temperature, the speed and direction of the wind, and the amount of humidity in the air.

Historical time-series data from PV production and its accompanying climatic factors form the basis of the PV power generation forecasting model. The data is separated into training and testing sets. Between training and testing, the dataset was divided from 75% to 25%. While training, we also validated our results against the testing set. However, we did the prediction procedure on both the training and testing sets for the final comparison.

Multiple data types from the above source were fed into our CNN network. No dedicated GPU is involved; the CPU processes the predictions. In a Google Colab setting, 104520 data is used to make test and training sets predictions. Our proposed system used around 13 GB of RAM and a Tesla T4 with approximately 15 GB of VRAM.

CHAPTER 6

INCORPORATION OF PHYSIOLOGICAL FEATURES IN DROWSINESS DETECTION USING DEEP NEURAL NETWORK APPROACH

6.1 Proposed Approach

Figs. 20 and 22 provide the proposed algorithm and architecture for drowsiness detection, respectively. The first stage is feeding video or image as an input which involves breaking down video frames from a fixed camera or a smartphone into a sequence of pictures. The driver's face is the main focal point in the video frames. The next is face detection which detects the face in the picture frames. In this case, we only noticed some part of the face. Instead, we only detected the eyes and mouth, as they are the main parameters for detecting drowsiness. One of the most used methods for detecting the driver's face in an image is Viola and Jones [146]. However, the entire picture is usually fed into the CNN with many filters, and features are extracted automatically. CNN integrates the images of eyes and mouth and pulls the feature. Eye and yawn posture is collected for feeding in CNN from these features. Attributes are often retrieved using methods like landmark localization [147], Histogram of Oriented Gradients (HOG) [148], and Local Binary Patterns (LBP) [149] in the face detection stage. The next step is feature analysis. Once features have been extracted, they can be further processed with Percentage of Eye Closure (PERCLOS) [150] or EAR for eye analysis or mouth-based algorithms for yawning detection. The final step is a classification that classifies the drowsiness level of a driver. If the weighted parameters identify tiredness, an alarm will sound, indicating that the driver should rest.

Images were given names according to their timestamps after being extracted from the sensor's data. To train our model, we employed a dataset of 500 pictures labeled in a separate CSV file with the image file's name, oxygen data, and heart rate data. Only 78 of the 500 photos represented sleepiness, whereas the remainder did not.

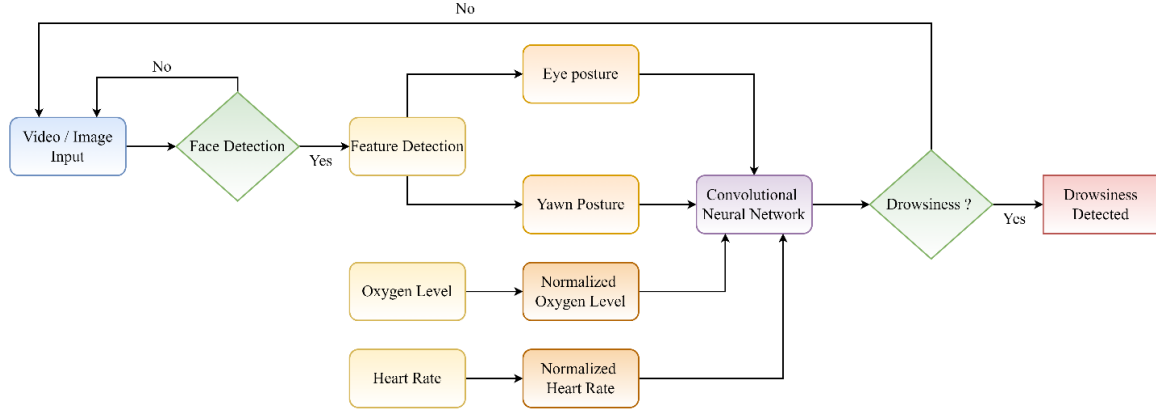


Fig. 20. Proposed Algorithm for Drowsiness Detection

We have used facial landmarks to pull out the images' left and right eyes and mouths. These modified versions of the original photos were stored in three directories with the same name as the original file. Afterward, the images were used to train the network. First, the pictures of the left eye, right eye, and mouth were each scaled down to 48×48 pixels and provided their own CNN layer. Next, the photos were processed using several parallel CNN, which resulted in the flattening of the max pool layers. Then, the normalized data on oxygen consumption and heart rate was appended to the characteristics that we had flattened. Finally, after being fed through dense and dropout layers, the concatenated pieces were sent to the output layer. The network configuration in our proposed CNN model is illustrated in Table 3.

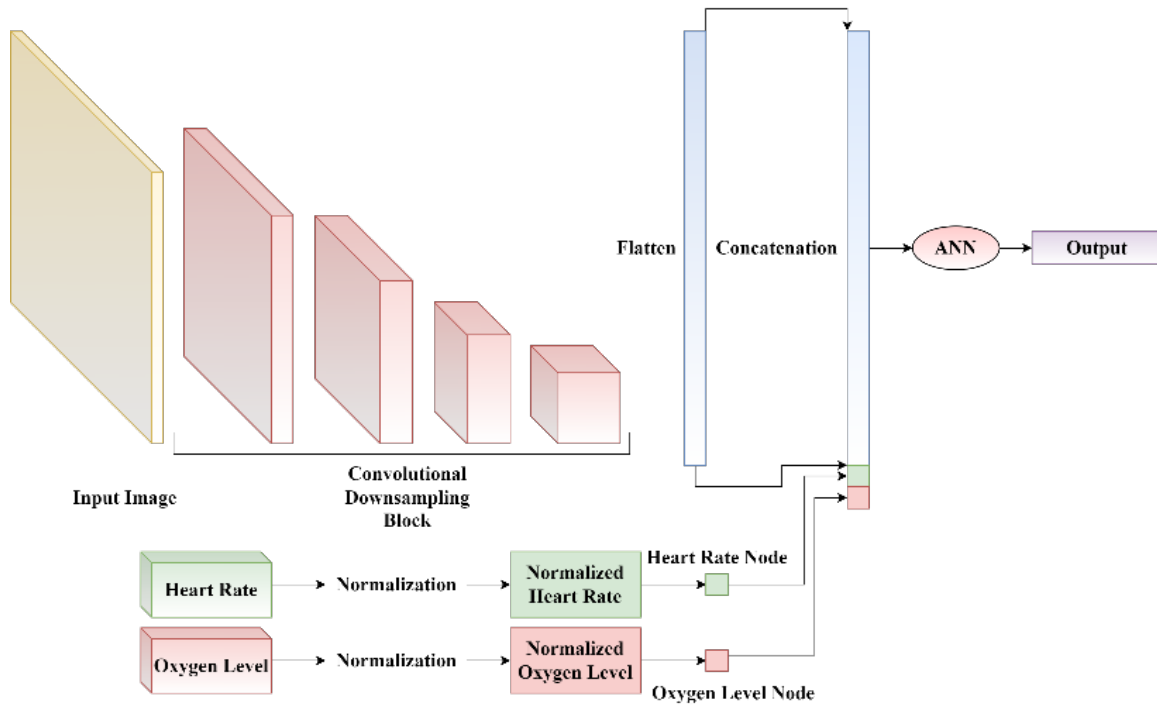


Fig. 21. Proposed Architecture of Drowsiness Detection

6.1.1 Dataset Description

Our data allows us to evaluate the proposed system’s effectiveness and thoroughly analyze its alternatives. The dataset was captured with a Raspberry Pi 4 IR-CUT night vision camera, which yielded images with a resolution of 1280×720 pixels. The camera also has infrared LEDs, so it can record the driver even if it’s nighttime. This camera has a 5MP sensor, a 3.6mm focal length that can be adjusted, and two LED lamps. With a frame rate of 30 fps, the video is relatively smooth. The purpose of this dataset is to model conditions that automobile drivers could face in the actual world. The dataset is composed of the training set and the validation set. In addition, sensors are installed in the car to simulate actual driving conditions. Figure 22 shows the architecture of our proposed method.

The sensor also measures heart rate in addition to pulse oximetry. This sensor has

low-noise circuitry with ambient light rejection in addition to LEDs, photodetectors, optical components, etc. The sensor provides an end-to-end system solution that simplifies the design process for portable and wearable gadgets. The sensor's LEDs run on 3.3V in addition to the 1.8V utilized by the rest of the device. Transmission takes place over an I2C-compatible interface, a standard in electronic hardware. When the program shuts down the module, it draws no standby current; therefore, the power rails can be on at all times.

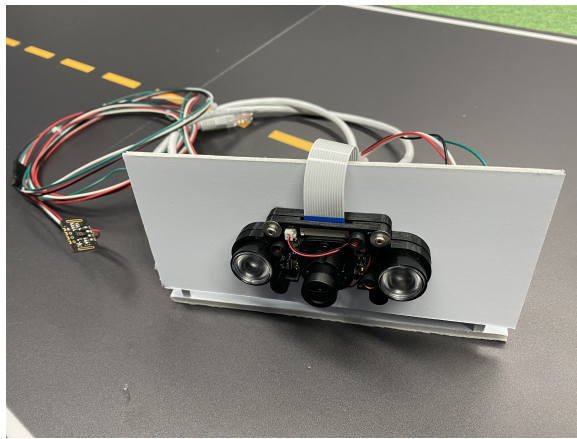


Fig. 22. Camera and Sensors Architecture

Table 3. Network configuration for the proposed CNN architecture

Layer	Output Shape	Parameter	Connected to
el_input	[(None, 48, 48, 1)]	0	[]
er_input	[(None, 48, 48, 1)]	0	[]
m_input	[(None, 48, 48, 1)]	0	[]
conv2d	(None, 48, 48, 8)	40	['el_input[0][0]']
conv2d_1	(None, 48, 48, 8)	40	['er_input[0][0]']
conv2d_2	(None, 48, 48, 8)	40	['m_input[0][0]']
max_pooling2d	(None, 24, 24, 8)	0	['conv2d[0][0]']
max_pooling2d_1	(None, 24, 24, 8)	0	['conv2d_1[0][0]']
max_pooling2d_2	(None, 24, 24, 8)	0	['conv2d_2[0][0]']
conv2d_3	(None, 12, 12, 16)	1168	['max_pooling2d[0][0]']
conv2d_4	(None, 12, 12, 16)	1168	['max_pooling2d_1[0][0]']
conv2d_5	(None, 12, 12, 16)	1168	['max_pooling2d_2[0][0]']
max_pooling2d_3	(None, 6, 6, 16)	0	['conv2d_3[0][0]']
max_pooling2d_4	(None, 6, 6, 16)	0	['conv2d_4[0][0]']
max_pooling2d_5	(None, 6, 6, 16)	0	['conv2d_5[0][0]']
flatten	(None, 576)	0	['max_pooling2d_3[0][0]']
flatten_1	(None, 576)	0	['max_pooling2d_4[0][0]']
flatten_2	(None, 576)	0	['max_pooling2d_5[0][0]']
oxygen	(None, 1)	0	[]
heart	(None, 1)	0	[]
Concatenate	(None, 1730)	0	['flatten[0][0]', 'flatten_1[0][0]', 'flatten_2[0][0]', 'oxygen[0][0]', 'heart[0][0]']
dense	(None, 128)	221568	['concatenate[0][0]']
dense_1	(None, 32)	4128	['dense[0][0]']
Output	(None, 1)	33	['dense_1[0][0]']

CHAPTER 7

AIR QUALITY PREDICTION USING DISTRIBUTED LSTM APPROACH FOR SMART CITY TESTBED

7.1 Air Quality Index (AQI)

The purpose of the air quality index is to serve as a forecasting tool, educating the public on the dangers of poor air quality and how to take precautions against them [151]. Five primary air pollutants were chosen for this purpose (CO, SO₂, PM₁₀, O₃, and NO₂); their concentrations were then categorized into six groups according to concentration breakpoints, and phrases defining the air quality were given to each group. For the AQI, the pollutant with the greatest concentration (or highest AQI value) is deemed the "responsible pollutant." It's essential to remember that the kind of people more vulnerable to pollution might vary widely. The most susceptible populations to air pollution depend on the pollutant in question; for example, those with lung problems, as well as youngsters and older individuals who are active outdoors, are more susceptible to the effects of ozone than others with heart disease [152] [153]. To quickly get the AQI for any given pollutant, we can use Equation 7.1:

$$I_p = \frac{I_{Hi} - I_{Lo}}{BP_{Hi} - BP_{Lo}}(C_p - BP_{Lo}) + I_{Lo} \quad (7.1)$$

In this equation, I_p is the pollutant index p , C_p is the rounded concentration of pollutant p , BP_{Hi} is the breakpoint that is greater than or equal to C_p , BP_{Lo} is the breakpoint that is greater than or equal to C_p , I_{Hi} is the AQI value associated

with BP_{Hi} , and I_{Lo} is the AQI value corresponding to BP_{Lo} [151] [152] [153]. The Air Quality index values is provided in Table Table 4 and the Pollutant-Specific Sub-indices for the Air Quality Index is provided in Table 5.

Table 4. Air Quality Index Categories [151]

AQI Values	Description	Color
0 to 50	Good	Green
51 to 100	Moderate	Yellow
101 to 150	Unhealthy for Sensitive Groups	Orange
151 to 200	Unhealthy	Red
201 to 250	Very Unhealthy	Purple
251 to 300	Hazardous	Maroon

Table 5. Pollutant-Specific Sub-indices for the Air Quality Index (AQI)[154]

AQI Categories	Ozone (ppm)		Particulate Matter ($\mu\text{g}/\text{m}^3$)		Carbon Monoxide (ppm)	Sulfur Dioxide (ppb)	Nitrogen Dioxide (ppb)
	[8-hour]	[1-hour]	[8-hour]	[1-hour]			
Good	0 - 0.054	-	0 - 12.0	0 - 54	0 - 4.4	0 - 35	0 - 53
Moderate	0.055 - 0.070	-	12.1 - 35.4	55 - 154	4.5 - 9.4	36 - 75	54 - 100
Unhealthy (SGroup)	0.071 - 0.085	0.125 - 0.164	35.5 - 55.4	155 - 254	9.5 - 12.4	76 - 185	101 - 360
Unhealthy	0.086 - 0.105	0.165 - 0.204	55.5 - 150.4	255 - 354	12.5 - 15.4	186 - 304	361 - 649
Very Unhealthy	0.106 - 0.200	0.205 - 0.404	150.5 - 250.4	355 - 424	15.5 - 30.4	305 - 604	650 - 1249
Hazardous	-	0.405 - 0.604	250.5 - 500.4	425 - 604	30.5 - 50.4	605 - 1004	1250 - 2049

7.2 Methodology

The complete process of our research is divided into two major parts - Training and Inference. As we had a pre-processed dataset, we didn't have to create or process the dataset. Instead, the dataset was divided into seven parts for different elements in the air. Then, we trained the model and made the inference using those data. Figure 23 provides the proposed architecture of our LSTM approach for predicting AQI for a smart city.

Figure 24 shows the working principle of this proposed approach. At first, we concatenated all our data into one dataset for seven elements. Then we preprocessed our data a little by deleting all the nan and null values. After that, we tried to encode our input dataset and divided the dataset into train and test parts. Finally, we normalized our dataset and trained it using our proposed model. After that, we classified our dataset based on the table and efficiently predicted the air quality index.

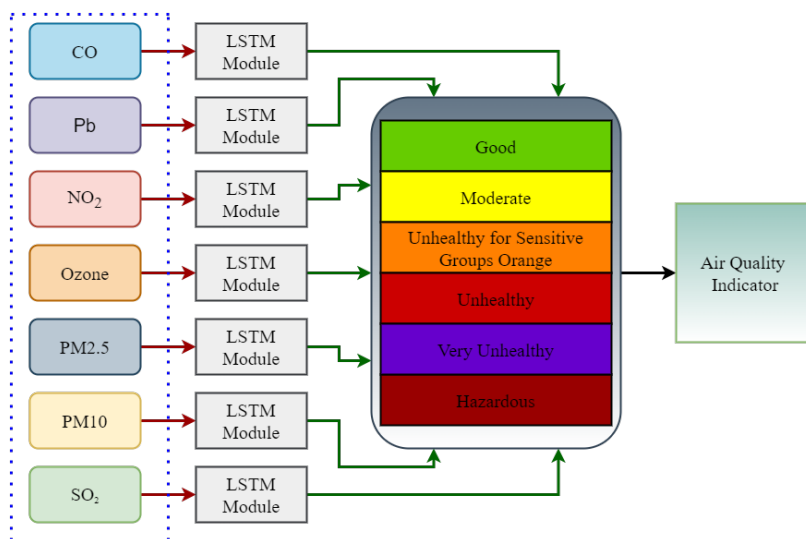


Fig. 23. The Proposed Distributed LSTM Architecture

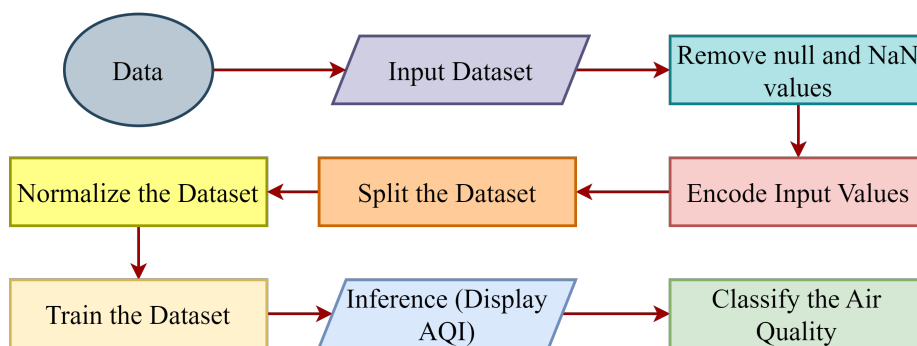


Fig. 24. The Proposed flow chart of the Suggested method

7.2.1 Training

We used an LSTM model built using 4 LSTM layers to train the dataset. Instead of making different models, the main goal was to create a single model architecture that would work with all seven different datasets for individual elements. But, after training the model with other datasets, the model was saved as seven additional files for all the features.

Table 6. Network Configuration for the proposed LSTM architecture

Layer	Output Shape	Parameter
lstm	(None, 300, 1, 1)	40800
lstm_1	(None, 300)	80400
lstm_2	(None, 300, 1)	80400
lstm_3	(None, 300, 50)	80400
dense	(None, 300, 50)	101

The model architecture for one LSTM module in our proposed LSTM model is provided in Table 6. Each dataset was trained for 25 epochs, as this produced the optimum results. After training, the saved models for each dataset were used to predict the value for both training and testing sets.

7.2.1.1 Forecasting

Using the inference process, we can visualize that the model can forecasting the future values for all the elements. To measure our model's performance, we have calculated the value of different performance metrics - MSE, MAE, and RMSE. This forecasting was later provided in the classification model to predict future air quality.

7.2.1.2 Classification

The mathematical data representation does not help us understand the air quality at a glance. To make the data more understandable and soothing, we classified the data into six different segments that show the air quality. This classification was based on Table 4 and Table 5. This helps us understand the predicted air quality more easily.

CHAPTER 8

ADAPTIVE CONTROL FOR SMART WATER DISTRIBUTION SYSTEMS

8.1 Testbed Design

Figure 25 depicts the water distribution system's simplified process flow. Each building has its water storage tank and variable frequency drive (VFD) pump to deliver water to each unit. One control valve is assigned to each floor, and each floor has two users.

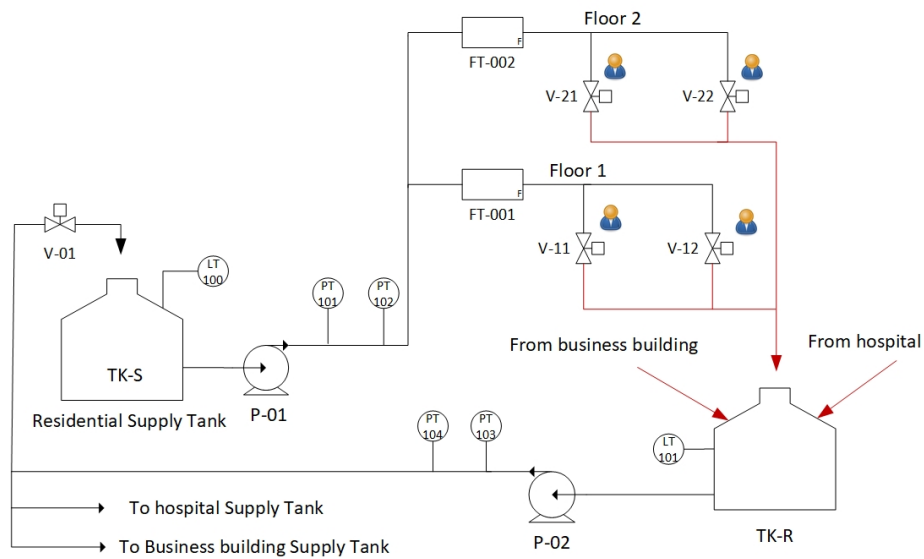


Fig. 25. A simplified process flow diagram for the water distribution system. Details for the business and hospital buildings are the same as the residential building and omitted to avoid cluttering the diagram.

The recycled water is collected in a tank that supplies all three structures. Each structure has a recycle pump to move water from the return tank to the storage tank.

When the reservoir level rises over a predetermined level or when the level in one or more supply tanks falls below a certain level, the reservoir pump activates and begins pumping water into the system.

8.1.1 Instrumentation and Control Design

The system pressure is regulated using a VFD pump. The recycle on/off pump controls the water level in the return and supply tank. Both pumps have an angular velocity sensor, a torque sensor, a temperature sensor, and an overpressure sensor. Water flow for each consumer is regulated using an automatic control valve representing user consumption. The control valves are equipped with position sensors to report the valve opening percentage. Each supply tank is fitted with an inlet on/off valve that closes when the water level exceeds a particular threshold value.

The water levels in the supply and return tanks are monitored using smart-level sensors. The flow on each floor is measured using a smart volumetric flow rate sensor. The total water consumption could be calculated either from the rate of change of supply tank level or by flow totalization of the flow sensors, assuming no system leakage. The discharge pressure of each pump is measured using a smart pressure sensor.

The general control philosophy is to regulate the system pressure or flow by adjusting the pump's angular velocity. The recycle pump starts when the return tank level is high and stops when the level is low. The inlet valve of each supply tank closes when the tank level is high. The water consumption represents a system disturbance and is simulated by developing a probabilistic model for the usage pattern and simulating the model to drive the consumption control valves.

8.1.2 Safety System Design

When the pressure at the pump’s discharge reaches a certain threshold, the VFD pump shuts off. For this, we employ a dedicated pressure sensor. In addition, if the tank’s water level drops too low, the pump will automatically shut off to prevent damage to the suction pump. The recycling pump’s security is maintained similarly.

8.1.3 Embedded System and Communication Architecture

The testbed uses a decentralized design in which smart sensors and actuators may have two-way communications without needing a central processing unit. Every actuator, like the variable frequency drive (VFD) pump, has its integrated system with algorithms for monitoring, control, and troubleshooting. The IEEE 802.11ac wireless standard exchanges data between sensors and actuators [155].

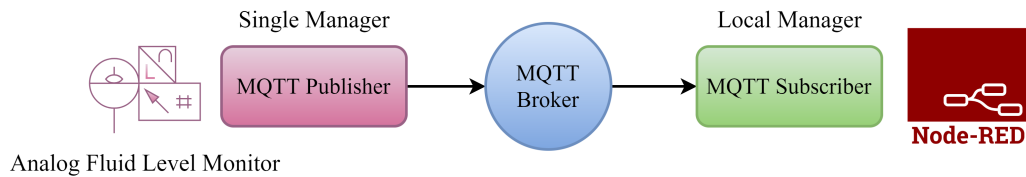


Fig. 26. MQTT (Message Queuing Telemetry Transport) communication for the testbed

The smart city infrastructure is communicated via MQTT (Message Queuing Telemetry Transport) [156]. MQTT has been selected to connect to various heterogeneous blocks in the more extensive smart city testbed at VCU. MQTT is a protocol developed especially for the "Machine-to-Machine" correspondence. The MQTT protocol operates over TCP (Transmission Control Protocol)/ IP (Internet Protocol), which can transmit data for different formats using a publisher/subscribers model instead of a client/server model. MQTT collects data from data publishers and

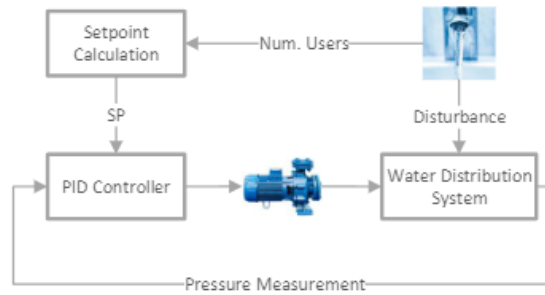


Fig. 27. Pressure control schemes for the water distribution system

transfers it to subscribers using a data broker [156]. The water distribution testbed sensors and actuators are connected to the broker via MQTT client software running on their respective Raspberry pi embedded boards. Figure 26 illustrates the MQTT communication architecture for the testbed.

8.2 Control System Design

Figure 27 illustrates in a single diagram the two control schemes proposed in the paper, and explained briefly in this section.

8.2.1 Single-loop Feedback Control

Single-loop pressure control, often known as the "classical" method, employs a proportional-integral-derivative (PID) controller to maintain constant pressure in the system. Due to its ease of use and little data needs, this control technique has widespread use. Based on the current pressure reading, the PID controller adjusts the VFD pump's voltage (or angular velocity in ideal drive versions). However, for most non-smart buildings already in existence, the user consumption profile—that is, the number of people using the system at once and the demand (system disturbance)—is unknown; therefore, the control strategy does not account for it.

8.2.2 Adaptive Feedback Control

By constantly adjusting the proportional, integral, and derivate (PID) controller, the pressure in a system is maintained at a predetermined value in the traditional single-loop pressure control technique. Because of its ease of use and low data requirements, this control method is widely implemented. The PID controller uses the current pressure reading to adjust the voltage provided to the VFD pump (or the angular velocity for perfect drive versions). Unfortunately, this control approach does not consider the user consumption profile, i.e., the number of concurrent users and the demand (system disturbance), as this data is often unavailable for conventional buildings. In this paper, the "Setpoint Calculation" in Figure 27 is only a lookup table correlating the number of active users with the desired temperature.

8.2.3 Disturbances and User Consumption Modeling

Since the user's engagement with the system might be seen as an external system disturbance, modeling the user's consumption profile is crucial for testing the resilience of any proposed control method. However, modeling the user's consuming behavior may be challenging due to the interdependency of timed user activities. Therefore, this study introduces a user-consumer model that abstracts away from complex interactions. Efforts to create more complex probabilistic models are constantly being investigated.

Figure 28's signal might represent actual user intake. The sign goes from low to high when the water tap is opened. The openness of a valve is indicated by its signal amplitude. The signal duration represents how long a user keeps the water faucet open. Finally, successive uses are captured by the time gap between two signal transitions. All the signal properties I've listed above could be more predictable,

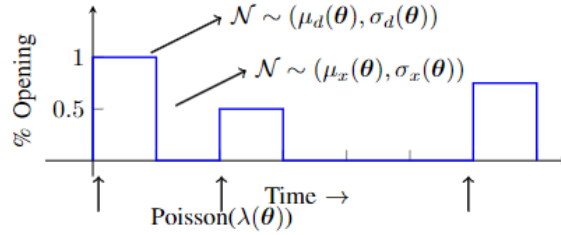


Fig. 4. A signal representation of user consumption.

Fig. 28. A signal representation of user consumption

making predicting user usage challenging. Furthermore, these features are interdependent. To recapitulate, we require a sophisticated probabilistic model to capture user behaviors.

We represent three different probability distributions to define the signal in Figure 28, parameterized by $\theta = [A \ E \ D \ I]$. The random variable representing the starting time of user consumption is modeled as a Poisson distribution with an arrival rate $\lambda(\theta)$. The random variable representing the duration of usage is modeled by a Gaussian distribution with parameters $\mathcal{N} \sim (\mu_d(\theta), \sigma_d(\theta))$. Similarly, the random variable representing the percentage opening of the faucet is modeled by a Gaussian distribution with parameters $\mathcal{N} \sim (\mu_x(\theta), \sigma_x(\theta))$. The three probability distributions and the user profile fully represent the user consumption behavior.

One of the key questions is how to estimate the model parameters $\lambda, \mu_d, \sigma_d, \mu_x, \sigma_x$ for each value of the parameter θ . The availability of a labeled user consumption dataset would enable us to verify the modeling assumption and estimate the parameter values. However, to our knowledge, there is no such detailed dataset. So in the meantime, we rely on our knowledge of common human practices for each user profile.

The model described here relies on two simplifications; one is that there is no dependence between usage distributions within and across time, and another is that

Table 7. User profile and time intervals, represented by the parameter vector θ

Attribute	Symbol	Possible Values
Age	A	Child, Adult, Elderly
Employment	E	No, Yes, Telework
Day	D	7 Days of the week
Time Interval	I	[6-9], [9-5], [5-9], [9-6]

There is no correlation between the times of arrival. Our probabilistic model takes into account user demographics as well as calendar and epoch variables. For example, a person’s age and employment position are two defining characteristics of their profile. In addition, the day is divided into four segments that correspond to the most common times of day when people utilize water. Table 7 summarizes the profile attributes and the possible values.

CHAPTER 9

SIMULATION RESULTS

9.1 Performance Evaluation

The Mean Absolute Error (MAE) measures how far off actual results are from projections. Like the RMSE metric equation, which quantifies the standard deviation of prediction errors, the MSE metric calculates the average square of the difference between the predicted and actual values. we can calculate MAE, MSE, RMSE using (3.8),(3.9),(3.10) respectively. [114].

$$MAE = \frac{1}{n} \sum_1^n |y - \hat{y}| \quad (9.1)$$

$$MSE = \frac{1}{n} \sum_1^n (y - \hat{y})^2 \quad (9.2)$$

$$RMSE = \sqrt{\frac{1}{n} \sum_1^n (y - \hat{y})} \quad (9.3)$$

In all above equations, y and \hat{y} denote the actual value and predicted value of PV generation, respectively.

Some mathematical variables are needed to understand the model's performance. Performance matrices are a common term for this type of metric. Being accurate is defined as the state of being precise or correct. Specificity refers to the ability to rule out false positives in detecting sleepiness in the driver, while sensitivity refers to the degree to which drowsiness is detected. In this study, the sensitivity, specificity, and accuracy are calculated by:

$$Sensitivity = \frac{TP}{(TP + FN)} \quad (9.4)$$

$$Specificity = \frac{TN}{(TN + FP)} \quad (9.5)$$

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (9.6)$$

$$Precision = \frac{(TP)}{(TP + FP)} \quad (9.7)$$

$$Recall = \frac{(TP)}{(TP + FN)} \quad (9.8)$$

where TP is the true positive detected conditions, TN is the true negative detected conditions, FP is the false positive detected conditions and FN is the false negative detected conditions.

The Precision of a system is defined as the fraction of relevant items among all items in a set. Thus, precision is a metric for evaluating the efficiency with which irrelevant samples are filtered out of the retrieved collection. Similar to the value placed on strong recall, high precision is ideally suited. An ideal classifier will have a precision and recall value of 1 [157, 158].

Table 8. Confusion Matrix

	Actual Class	
Predicted Class	True Positive (TP)	False Positive (FP)
	False Negative (FN)	True Negative (TN)

A confusion matrix is a metric for evaluating the efficacy of machine learning solutions to classification problems with multi-class outcomes. It's a table showing four distinct permutations of expected and actual numbers. It calculates Area Under the Curve (AUC) and other ROC measures [159]. Table 8 provides the concept of a

confusion matrix.

9.2 Smart Energy

Figure 29 shows the results of predicting PV generation using our proposed approach over the whole simulation period. Three trajectories, actual, training, and testing data, are illustrated; i.e., the dataset is divided into training and testing datasets (75% training data and 25% testing data). Besides, for a clear illustration, prediction outputs of the proposed method are shown in three different time sequence ranges; 9000 – 10000, 76000 – 81000, and 99000 – 100000, in Figure 30, Figure 31, and Figure 32, respectively. According to the graphs, the prediction performance of the proposed method is satisfying; the actual values and predicted data converge in shorter periods.

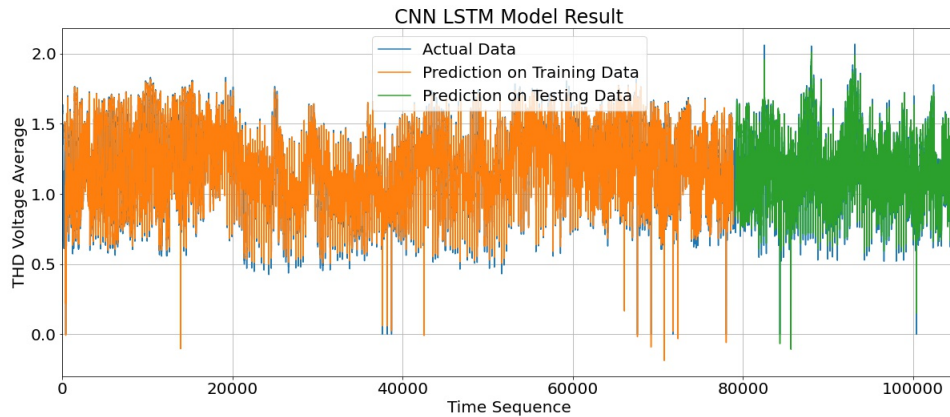


Fig. 29. Prediction Comparison using CNN-LSTM over the whole learning period

The results of learning PV power production through the LSTM approach (baseline) are presented in Figure 35. For the sake of clarity, Figure 36, Figure 37, and Figure 38 show the forecasting results in three different time periods, 9000 – 10000,

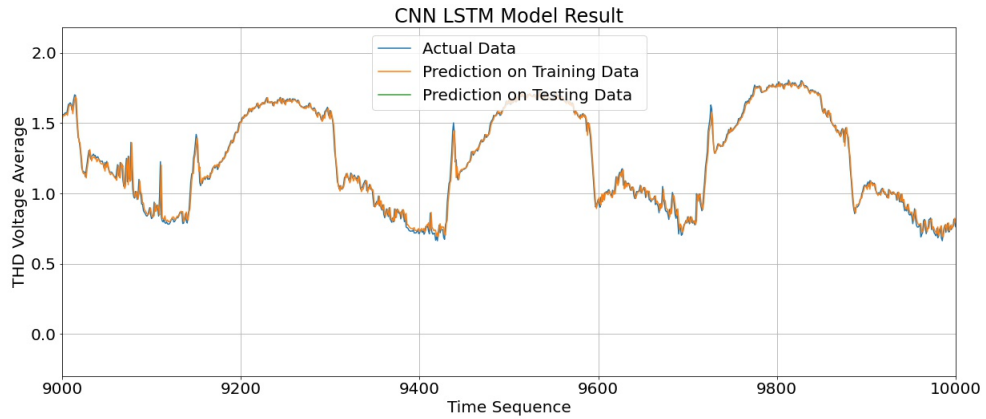


Fig. 30. Prediction Comparison using CNN-LSTM over the time sequence 9000-10000

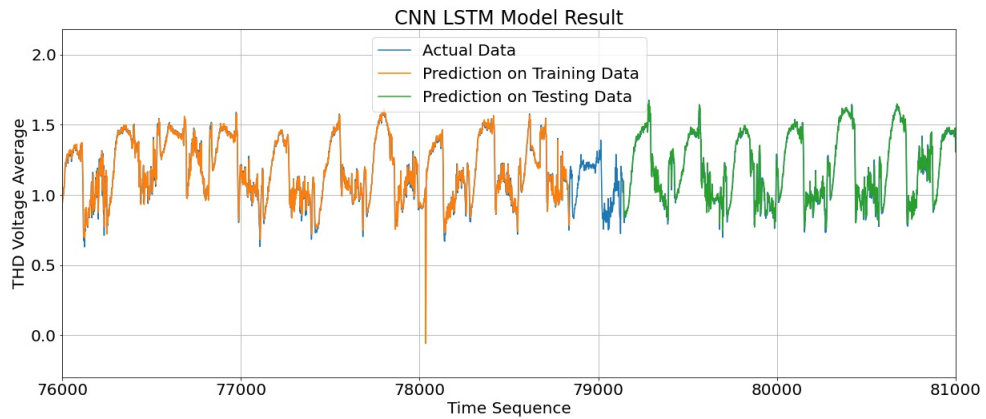


Fig. 31. Prediction Comparison using CNN-LSTM over the time sequence 76000-81000

76000 – 81000, and 99000 – 100000, respectively.

Comparing the results from the two approaches, predictions using the hybrid approach are slightly closer to the actual data; however, the results of the baseline approach are still reliable. The RMSE, MAE, and MSE error values and the computational overhead for the two methods (hybrid CNN-LSTM and LSTM models) are provided in Table 9. According to the error values, the accuracy of our proposed network (through both training and testing data) is significantly better than the baseline

method (single LSTM model).

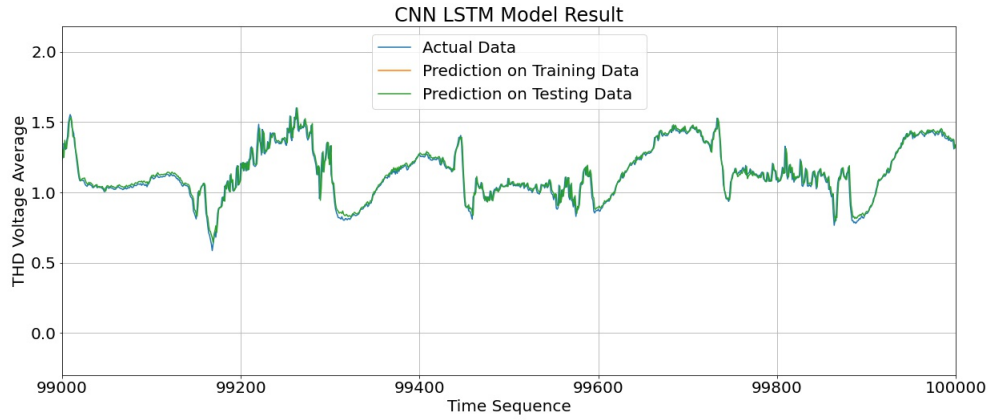


Fig. 32. Prediction Comparison using CNN-LSTM over the time sequence 99000-100000

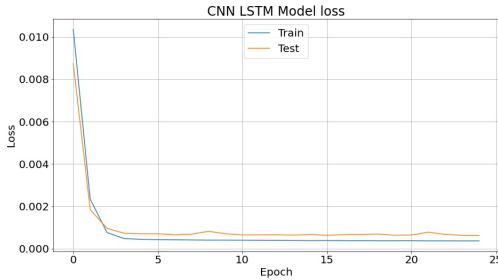


Fig. 33. Training and testing data loss value using proposed method

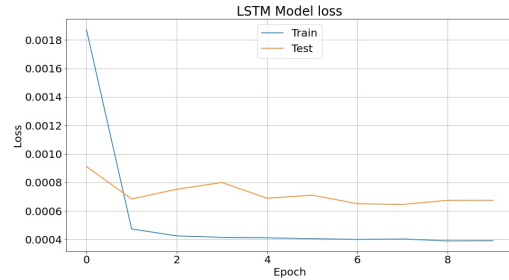


Fig. 34. Training and testing data loss values using baseline method

However, the system memory usage and execution time are significantly higher in our proposed prediction method compared to the baseline. The higher computational overhead in the proposed method is expected, considering that 13 different types of input datasets are used in the CNN-LSTM model to carry out the predictions. In contrast, only one input data is used in the LSTM model. Besides, the CNN-LSTM model has to perform complex calculations of the CNN layer. All the 104520 data points (with a 5-minute sampling time) are being analyzed simultaneously. The la-

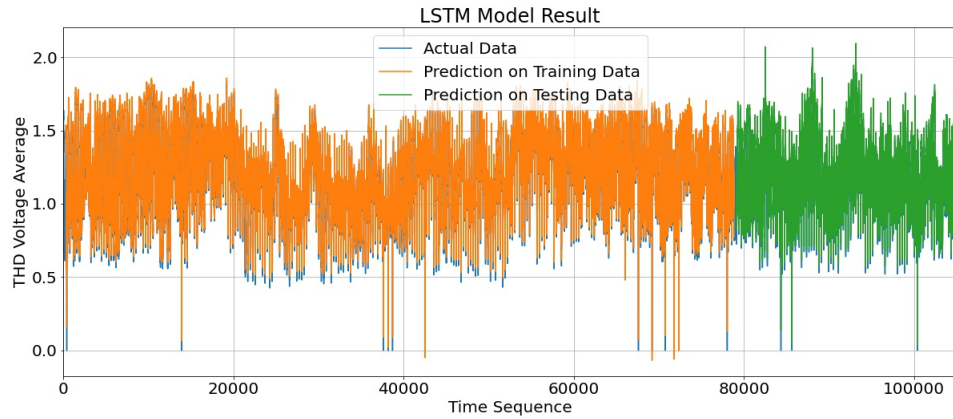


Fig. 35. Prediction Comparison using LSTM over the whole learning period

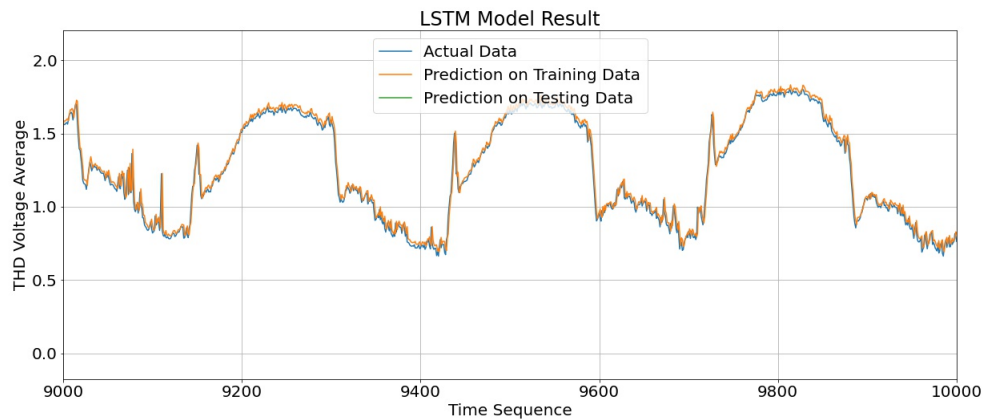


Fig. 36. Prediction Comparison using LSTM over the time sequence 9000-1000

tency and computation complexity of CNN-LSTM is slightly higher than a single LSTM approach. For instance, the average time to train each epoch in LSTM was 60-80ms/step, whereas this value was measured at 220-250ms/step for CNN-LSTM. However, any modern computer can handle the computation complexity of our proposed hybrid model, and it is not considered a critical issue.

Figure 33 and Figure 34 present the loss value minimization for the hybrid CNN-LSTM and baseline architectures, respectively. Comparing the two figures, it

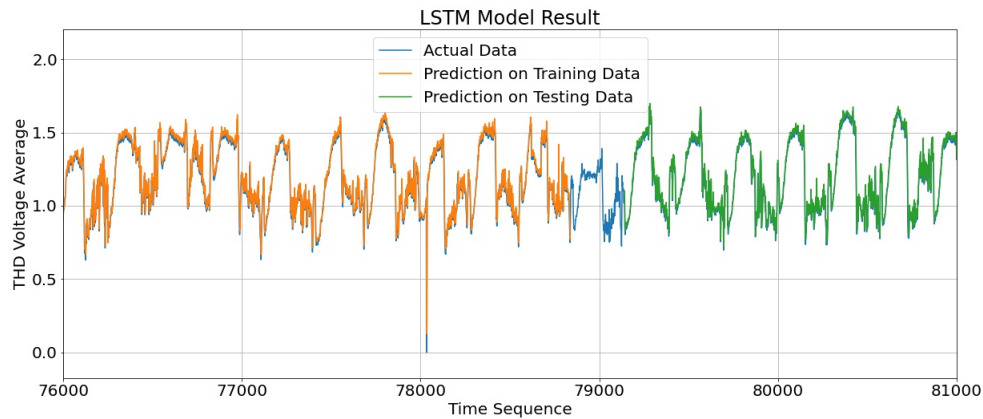


Fig. 37. Prediction Comparison using LSTM over the time sequence 76000-81000

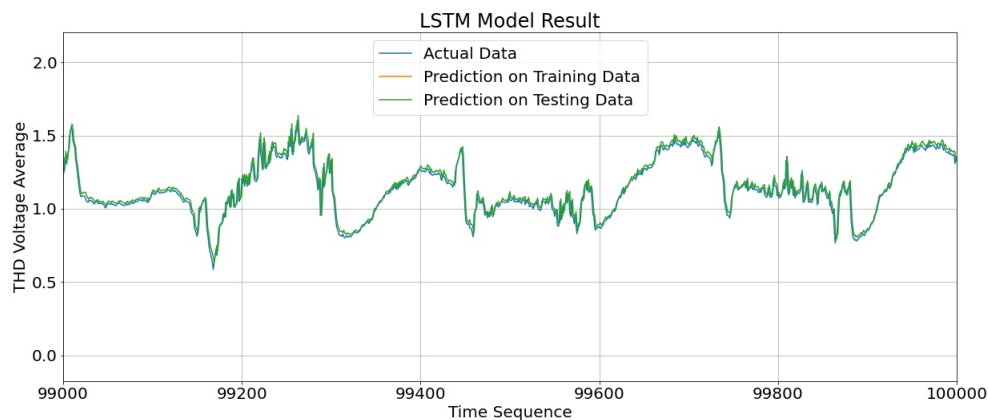


Fig. 38. Prediction Comparison using LSTM over the time sequence 99000-100000

can be observed that both models' fitting happens roughly simultaneously. According to the curves in Figure 33, overfitting or underfitting did not occur in the network, and the model functions with sufficient accuracy within 20 epochs. However, the baseline model (single LSTM) shows overfitting; since the training loss trajectory continues to decrease with experience, and the test data loss drops to a point and begins increasing again.

Table 9. Performance comparison of the proposed approach and the baseline

Prediction Method	MSE		MAE		RMSE		System Memory Usage (%)	GPU Memory Usage (%)	Time (s)
	Train	Test	Train	Test	Train	Test			
LSTM	0.0022	0.0037	0.0320	0.0392	0.0477	0.0610	16.1	4.18	75
CNN-LSTM	0.0017	0.0031	0.0254	0.0351	0.0420	0.0564	54.7	4.09	253
BiLSTM	0.395	0.0604	0.1684	0.1845	0.1987	0.1987	15.4	6.44	393

9.3 Smart Transportation System

80% of the training validation set is used for training purposes. In contrast, the remaining 20% is used for validation purposes. Figure 39, and Figure 40 present the model accuracy and loss of the proposed CNN architectures, respectively. Training loss is lesser than the testing loss in Figure 40. Figure 39 shows that the training and testing accuracy are around 1 and 0.6. The reason for getting lesser testing accuracy is the random shuffling of the same types of images. Testing data had more drowsy images, which incurred lesser accuracy.

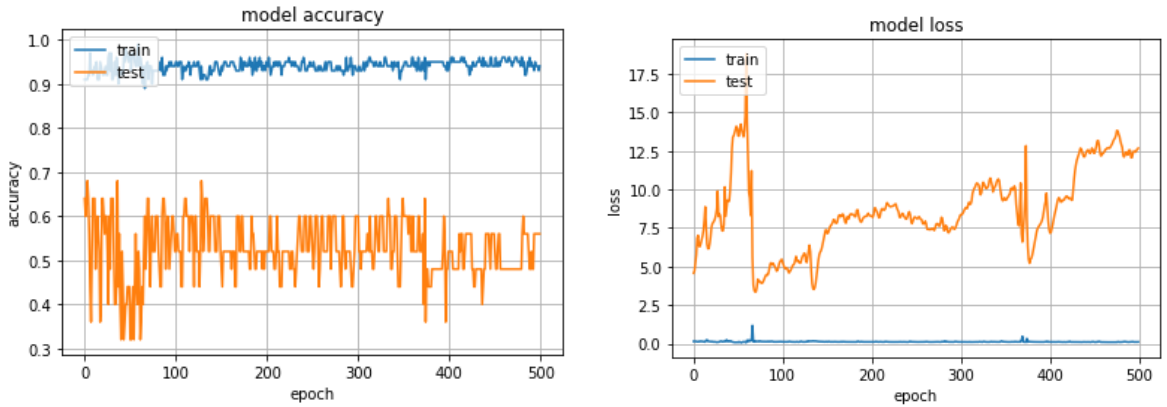


Fig. 39. Training and testing data accuracy using the proposed method Fig. 40. Training and testing data loss values using the proposed method

Figure 40 shows that the testing loss is close to zero, and training loss has erratic values. This happens because of our sensor’s sensitivity in our experiment. The sensor is too much sensitive to use. We try to replicate the images with data

while driving, which may cause a higher loss in training. That is why the loss is slightly higher than the standard value. Figure 41 shows the confusion matrix for our proposed approach.

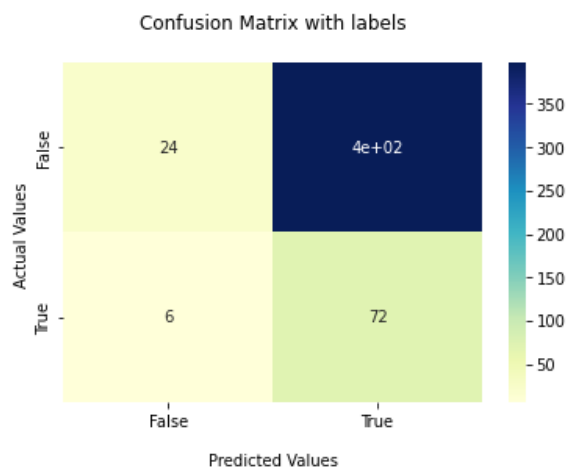


Fig. 41. Confusion matrix of the proposed approach

Table 10. Results for the proposed methodology

Performance Metrics	Value (%)
Accuracy	94
Misclassification Rate	6
Precision	72.92
Sensitivity	75
Specificity	98.51

9.4 Smart Environment

From Figure 42 to Figure 47 shows the result for CO, SO₂, PM₁₀, O₃, and NO₂ accordingly. The graphs show the inference result for both the training and testing dataset. The low MSE, MAE, and RMSE values help us conclude that the model

designed for all the datasets is efficient and works well to predict the future air quality for a given area.

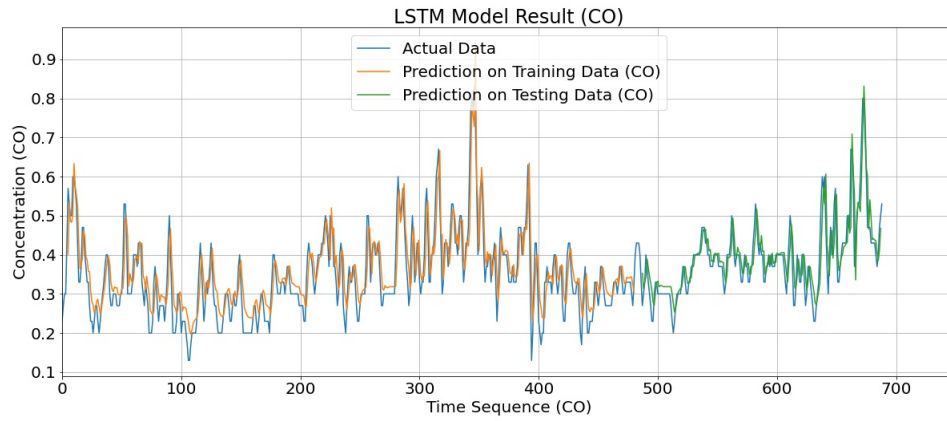


Fig. 42. Forecasting Result for CO

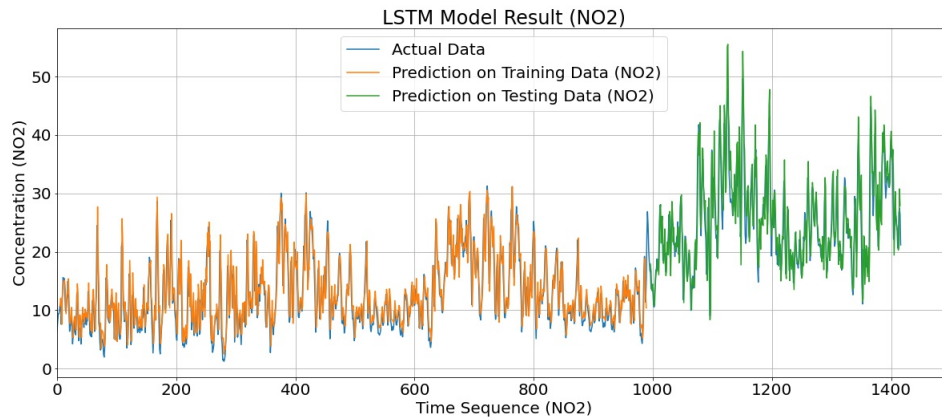


Fig. 43. Forecasting Result for NO₂

9.5 Smart Water Distribution System

We simulated the system for six different usage patterns; one active user (1), two users on the same floor (2-S), two users on opposite floors (2-O), three users with the

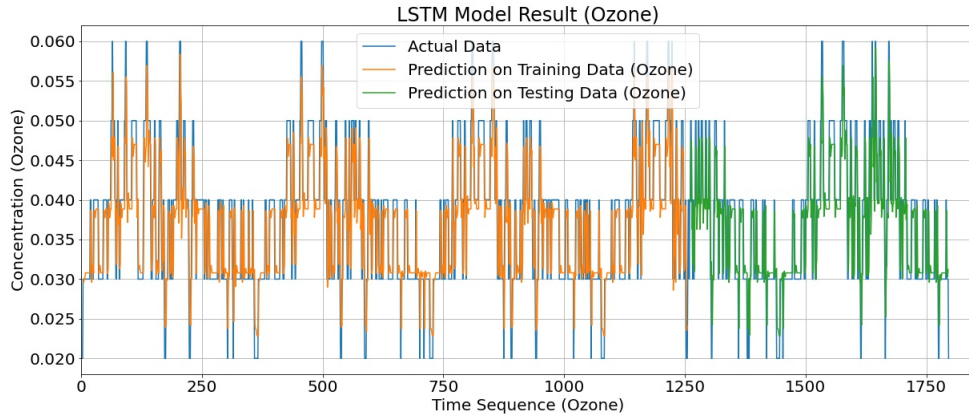


Fig. 44. Forecasting Result for Ozone

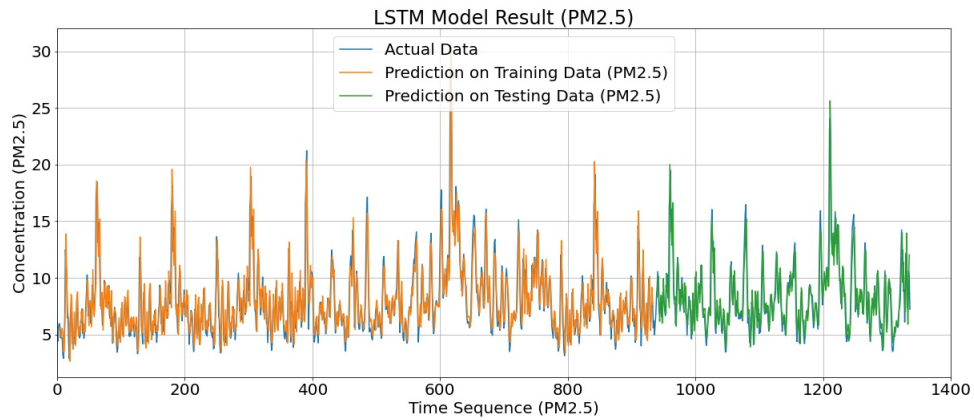


Fig. 45. Forecasting Result for $PM_{2.5}$

additional two users on the same floor (3-S), three users with the other two users on alternative floors (3-O), and all users (4). For this experiment, the performance is measured by the deviation from a reference flow rate (user convenience) and power consumption.

Figure 49 illustrates the pressure profile for the two control schemes. The fixed control scheme maintained the system pressure at the desired setpoint as expected. The dynamic pressure shows the required pressure setpoint for each usage pattern.

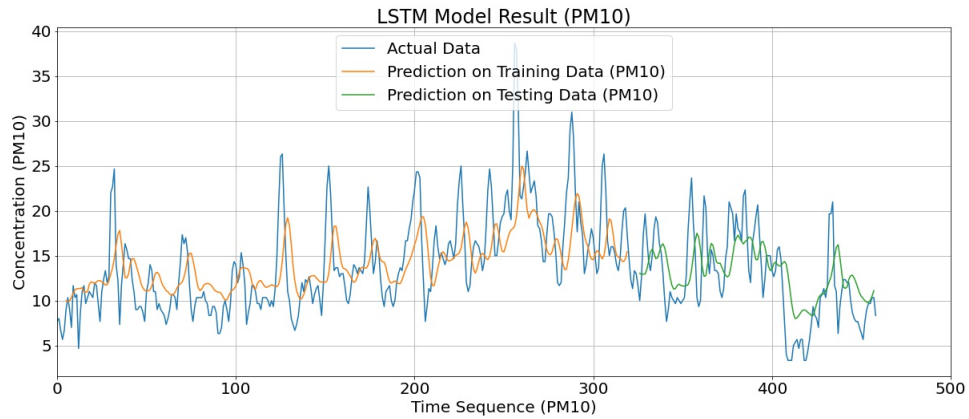


Fig. 46. Forecasting Result for PM₁₀

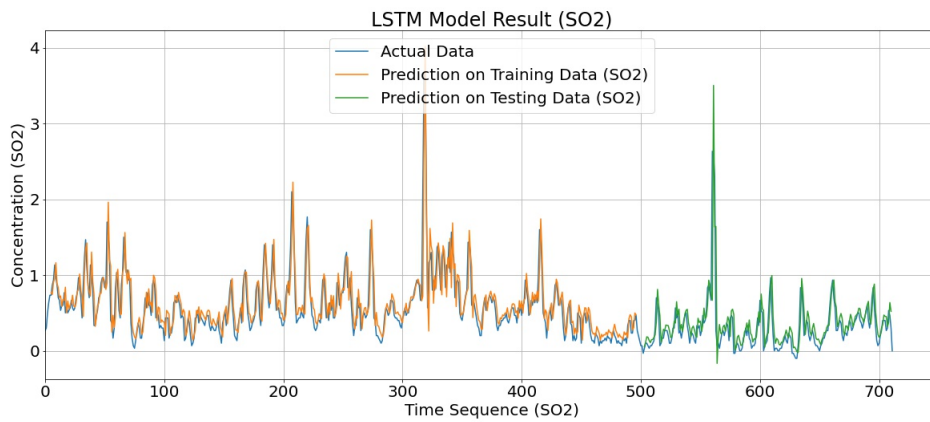


Fig. 47. Forecasting Result for SO₂

The PID controller kept the setpoint needed, so the setpoint and actual pressure measurement were the same.

Figure 50 shows the flow rate for the pivot user (the user is always on in all usage patterns). The fixed pressure control scheme results in a wide variation of the flow rate, hence a negative user experience. For a single user, the pressure head is so high that it doubles the required flow rate. For the other usage patterns, the flow rate oscillates based on other users' locations. Reducing the pressure setpoint will

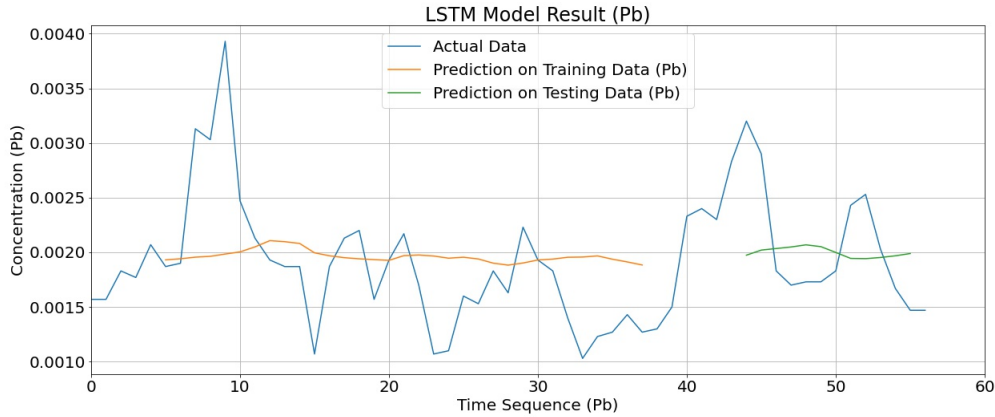


Fig. 48. Forecasting Result for Pb

shift the whole curve down, resulting in an underflow for the user when other users are online instead of an overflow condition. The adaptive control scheme reduces the variation in the flow rate, although not perfectly. The figure also shows that a lower-floor user will always have a higher flow rate regardless of the usage pattern. This is not surprising since the pressure loss will be higher on higher floors. The figure shows a fundamental limitation of the current control scheme, where we only have one degree of freedom. We can control only the pump speed, so theoretically, it is impossible to maintain the flow for all users with a single manipulated variable.

Figure 51 shows the power consumption per user. The power consumption is reduced significantly using the adaptive control scheme. The main reason is that fixed pressure control results in over-pressure for most scenarios that are not needed to achieve the desired flow rate; hence excessive power is unnecessarily spent.

9.5.1 Discussion

From the experimental results, we conclude that adaptive feedback control utilizing IoT data results in less flow rate variability (user convenience) and less power

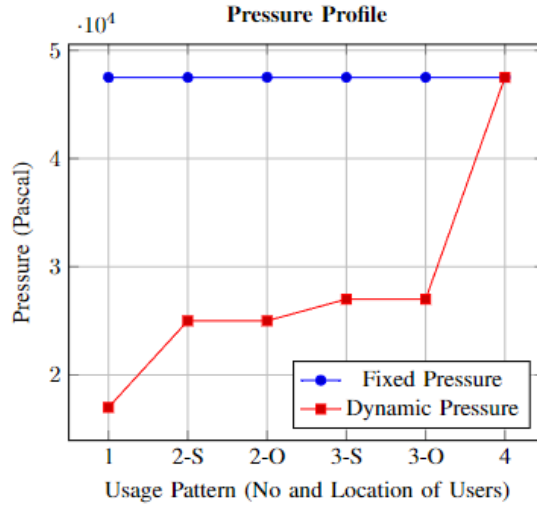


Fig. 49. Pressure profile for the two experimental control schemes vs different usage patterns. For the usage patterns, "S" is for the Same floor, and "O" is for the Opposite floor. The fixed pressure scheme utilizes one pressure setpoint for all usage patterns, while the dynamic pressure scheme varies the pressure set point according to the usage pattern.

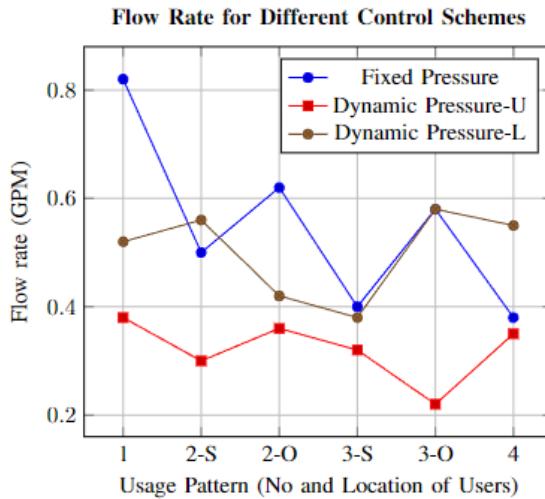


Fig. 50. Flow rate variation for the two pressure control schemes for a fixed user. For the dynamic pressure scheme, the flow rate for two different users at two different floors is shown. "U" stands for the Upper floor, and "L" for the Lower floor.

Table 11. Performance comparison of the proposed approach and the baseline

LSTM	MSE		MAE		RMSE		System Memory Usage (%)
	Train	Test	Train	Test	Train	Test	
CO	0.003	0.002	0.0431	0.0343	0.0552	0.0474	18.9
PM _{2.5}	2.340	2.034	1.111	1.033	1.529	1.426	20.7
PM ₁₀	19.605	16.807	3.225	3.187	4.427	4.099	19.1
NO ₂	6.434	11.935	1.953	2.676	2.536	3.454	19.1
SO ₂	0.044	0.055	0.148	0.134	0.210	0.235	16.5
O ₃	2.354	2.540	0.003	0.003	0.004	0.005	18.8
Pb	3.830	3.027	0.0004	0.0004	0.0006	0.0005	18.8

Table 12. Performance comparison of the proposed approach and the baseline

Elements	Train		Test	
	Precision	Recall	Precision	Recall
CO	1	1	1	1
PM _{2.5}	0.98	0.97	0.99	0.97
PM ₁₀	1	1	1	1
NO ₂	1	1	1	1
SO ₂	1	1	0.98	0.98
O ₃	1	0.99	1	0.99
Pb	1	1	1	1

consumption (cost-effective operation). However, tight system control for a more efficient process is impossible with a single manipulated variable.

This work aims to highlight the power of utilizing the data provided by the IoT echo system, even with very limited system controllability. However, IoT provides opportunities for optimal system operation beyond what we showed here and without the need for additional investment. As an example, for smart valves that could be regulated remotely, the user could specify the required flow rate using a simple digital

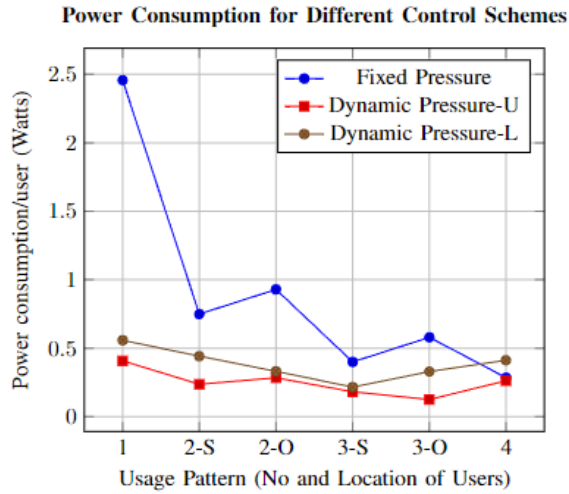


Fig. 51. Power consumption per user for the two pressure control schemes.

display, possibly from among preset values, and this information represents the desired setpoint for the controller. The controller then could adjust multiple inputs, including pump speed, user valve percentage opening, and other system components, to achieve the desired user setpoint. Although it may seem counter-intuitive for the first time that the user valve is adjusted by a controller and not by the user, as we used to, what we do at home or the office is that we change the valve opening to achieve the flow rate we need. Therefore, a manual adjustment could be eliminated or kept to a minimum in a smart building setup. Furthermore, this higher degree of control would allow us to design more sophisticated optimal control schemes that minimize energy usage and water consumption while maintaining minimum user convenience.

CHAPTER 10

CONCLUSIONS AND FUTURE WORKS

The notion of a "Smart City" was inspired by the necessity of finding practical solutions to widespread urban issues. It seeks to do so by enhancing the quality of life in existing settlements through the strategic deployment of hardware and software [160]. Energy, transportation, water, public safety, and other vital service sectors within the smart city are controlled in such a way as to assure they are correct operation, with consideration given to the upkeep of a clean, inexpensive, and secure environment in which inhabitants may live comfortably [161]. Its energy infrastructure stands out as an absolute need, considering a city's dependence on its other critical infrastructure [162].

10.1 Smart Energy

An uncertainty-aware prediction model based on deep learning is proposed for the PV generation forecasting problem. First, a convolutional neural network (CNN) is developed to capture local features from the data. In the next step, LSTM neural network extracts temporal relationships and generates PV output forecasts depending on the dataset's features. We obtained a highly accurate and flexible prediction model by integrating these two techniques. The proposed approach is applied to a real dataset (DKASC), and its performance is validated for various forecasting horizons. The performance of our proposed deep learning model is compared to that of a single LSTM and a BiLSTM model (which are known to be robust, efficient, and accurate learning approaches). Simulation results proved that the proposed hy-

brid CNN-LSTM model outperforms the baseline approach regarding accuracy and training/testing fitting.

10.2 Smart Transportation System

This paper proposes an algorithm for driver drowsiness detection using a convolutional neural network and physiological approaches. A new perspective towards driver drowsiness detection is presented by integrating deep neural network approaches and the heart rate and blood oxygen level to detect, analyze, and monitor a driver's drowsiness while driving. Previous methods could only make decisions based on features such as eye blinks, eye closure, forehead strain marks, or even eyebrow shapes. Other modern approaches were based on carefully hand-engineered features detecting driver drowsiness based on human facial expressions. The convolutional neural networks-based representation feature learning approach provides an automated and efficient set of features that can accurately classify the driver as drowsy or non-drowsy. Integration between those aspects is a new concept to make the proposed techniques more robust, valuable, and efficient. The proposed approach is used for an actual dataset made by ourselves, and its performance is validated and calculated by different performance matrices. Simulation results provided different horizons for integrating various physiological features with the proposed CNN model. As a future work, We will use this module as part of experiments in our smart city platform [19].

10.3 Smart Environment

Despite the detrimental effects of air pollution, it is vital to have a reliable model for predicting AQI levels to improve urban public health and foster sustained social progress. Because of the direct influence on city administration and citizen health that air pollution forecasts have in densely populated places, such knowledge is of

paramount relevance. It is essential to use deep learning neural networks like the long short-term memory (LSTM) to assist in policy making that prioritizes improving air quality To create more sustainable cities. The suggested method may predict future AQI values by studying existing data on CO, SO₂, PM₁₀, O₃, and NO₂ concentrations. Richmond, Virginia, USA, is the location for this data set's creation. Results from the current study show that the distributed LSTM technique outperforms standalone neural network predictors and regression models in predicting the AQI. the sample size indicates only a small time frame (one day). A more rigorous statistical analysis and more definitive conclusions might have been achieved with an extended period of hourly data. The capacity of any model to predict the future may vary depending on the length of time spent anticipating or the size of the steps used to advance the prediction. Second, additional aspects, such as the city's economy and traffic flow, might be considered in future studies if they haven't already been. This research presented a practical approach to determining the concentration values of various contaminants contributing to pollution and combining them to generate AQI values for that period. Multiple weather stations and user apps can utilize the proposed model to provide instantaneous predictions of the pollution level.

10.4 Smart Water Distribution System

We presented the design of a laboratory testbed for a water distribution system that is part of a larger smart city project. We built a simulator for the physical system hydraulics to use it for model-based design of the plant control system. We showed by numerical simulations that utilizing the information provided by the IoT infrastructure results in a convenient energy-efficient operation for the system without additional cost. This is a work in progress and several extension are currently ongoing. We will be building a state space model for the complete water distribution system

for model-based control design. We are in the process of building the real testbed and will utilize it to collect real time data and adjust the mathematical model using system identification techniques. Both the testbed and the simulator will have an expanded number of buildings and users to mimic real situations. More complex control algorithms that leverage IoT data are currently under study, including optimal flow control by varying individual user valves under water and energy consumption as well as user convenience constraints. We have faced the main challenges in this work: scalability and adaptability because every system has its parameters and variables. To sync with those parameters and variables, we have to execute the whole control method, simulation, and the proposed testbed properly. Finally, more realistic user consumption profiles will be used in system analysis and design.

Appendix A

ABBREVIATIONS

ICT	Information and Communication Technology
IoT	Internet of Things
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
ANN	Artificial Neural Network
PV	Photovoltaic Energy
ITS	Intelligent Transportation System
MLP	Multilayer Perceptron
RBF	Radial Basis Function
SMLP	Square Multilayer Perceptron
WDS	Smart Water Distribution System
TL	Transfer Learning
CNN	Convolutional Neural Network
GAP	Global Average Pooling
LSTM	Long Short Term Unit
ReLU	Rectified Linear Unit
SVM	Support Vector Machine
BGD	Batch Gradient Descent
SGD	Stochastic Gradient Descent
AQI	Air Quality Index

REFERENCES

- [1] Edward O'Dwyer et al. "Smart energy systems for sustainable smart cities: Current developments, trends and future directions". In: *Applied energy* 237 (2019), pp. 581–597.
- [2] Yi Liu et al. "Intelligent edge computing for IoT-based energy management in smart cities". In: *IEEE network* 33.2 (2019), pp. 111–117.
- [3] Riccardo Petrolo, Valeria Loscri, and Nathalie Mitton. "Towards a smart city based on cloud of things, a survey on the smart city vision and paradigms". In: *Transactions on emerging telecommunications technologies* 28.1 (2017), e2931.
- [4] Unai Aguilera et al. "Citizen-centric data services for smarter cities". In: *Future Generation Computer Systems* 76 (2017), pp. 234–247.
- [5] Paolo Neirotti et al. "Current trends in Smart City initiatives: Some stylised facts". In: *Cities* 38 (2014), pp. 25–36.
- [6] Zaib Ullah et al. "Applications of artificial intelligence and machine learning in smart cities". In: *Computer Communications* 154 (2020), pp. 313–323.
- [7] Fadi Al-Turjman. "Information-centric framework for the Internet of Things (IoT): Traffic modeling & optimization". In: *Future Generation Computer Systems* 80 (2018), pp. 63–75.
- [8] Zaheer Allam and Zaynah A Dhunny. "On big data, artificial intelligence and smart cities". In: *Cities* 89 (2019), pp. 80–91.

- [9] Hongjia Li et al. “Deep reinforcement learning: Framework, applications, and embedded implementations”. In: *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE. 2017, pp. 847–854.
- [10] Reinaldo Padilha França et al. “An overview of the machine learning applied in smart cities”. In: *Smart cities: A data analytics perspective (2021)*, pp. 91–111.
- [11] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.
- [12] Sendhil Mullainathan and Jann Spiess. “Machine learning: an applied econometric approach”. In: *Journal of Economic Perspectives* 31.2 (2017), pp. 87–106.
- [13] M Hadi Amini, Javad Mohammadi, and Soumya Kar. “Promises of fully distributed optimization for iot-based smart city infrastructures”. In: *Optimization, Learning, and Control for Interdependent Complex Networks*. Springer, 2020, pp. 15–35.
- [14] Federico Montori, Luca Bedogni, and Luciano Bononi. “A collaborative internet of things architecture for smart cities and environmental monitoring”. In: *IEEE Internet of Things Journal* 5.2 (2017), pp. 592–605.
- [15] Farid Ghareh Mohammadi et al. “Data analytics for smart cities: Challenges and promises”. In: *Cyberphysical Smart Cities Infrastructures: Optimal Operation and Intelligent Decision Making (2022)*, pp. 13–27.
- [16] Farid Ghareh Mohammadi, M Hadi Amini, and Hamid R Arabnia. “An introduction to advanced machine learning: meta-learning algorithms, applications, and promises”. In: *Optimization, Learning, and Control for Interdependent Complex Networks*. Springer, 2020, pp. 129–144.

- [17] Fátima Trindade Neves, Miguel de Castro Neto, and Manuela Aparicio. “The impacts of open data initiatives on smart cities: A framework for evaluation and monitoring”. In: *Cities* 106 (2020), p. 102860.
- [18] “Desert Knowledge Precinct in Central Australia”. In: (). URL: <http://dkasolarcentre.com.au/download?location=alice-spring>.
- [19] Nasibeh Zohrabi et al. “OpenCity: An Open Architecture Testbed for Smart Cities”. In: *2021 IEEE International Smart Cities Conference (ISC2)*. 2021, pp. 1–7. DOI: 10.1109/ISC253183.2021.9562813.
- [20] M. Alrashidi et al. “Short-Term PV Output Forecasts with Support Vector Regression Optimized by Cuckoo Search and Differential Evolution Algorithms”. In: *2018 IEEE International Smart Cities Conference (ISC2)*. 2018, pp. 1–8. DOI: 10.1109/ISC2.2018.8656685.
- [21] Yuchi Sun, Vignesh Venugopal, and Adam R. Brandt. “Short-term solar power forecast with deep learning: Exploring optimal input and output configuration”. In: *Solar Energy* 188 (2019), pp. 730–741. ISSN: 0038-092X. DOI: <https://doi.org/10.1016/j.solener.2019.06.041>. URL: <https://www.sciencedirect.com/science/article/pii/S0038092X19306164>.
- [22] A. Ryu et al. “Preliminary Analysis of Short-term Solar Irradiance Forecasting by using Total-sky Imager and Convolutional Neural Network”. In: *2019 IEEE PES GTD Grand International Conference and Exposition Asia (GTD Asia)*. 2019, pp. 627–631. DOI: 10.1109/GTDAsia.2019.8715984.
- [23] Na Dong et al. “A novel convolutional neural network framework based solar irradiance prediction method”. In: *International Journal of Electrical Power & Energy Systems* 114 (2020), p. 105411. ISSN: 0142-0615. DOI: <https://doi.org/10.1016/j.ijepes.2020.105411>.

org/10.1016/j.ijepes.2019.105411. URL: <https://www.sciencedirect.com/science/article/pii/S0142061518332915>.

- [24] Biaowei Chen et al. “Very-short-term power prediction for PV power plants using a simple and effective RCC-LSTM model based on short term multivariate historical datasets”. In: *Electronics* 9.2 (2020), p. 289.
- [25] Donghun Lee and Kwanho Kim. “Recurrent Neural Network-Based Hourly Prediction of Photovoltaic Power Output Using Meteorological Information”. In: *Energies* 12.2 (2019). ISSN: 1996-1073. DOI: 10.3390/en12020215. URL: <https://www.mdpi.com/1996-1073/12/2/215>.
- [26] Fei Wang et al. “A day-ahead PV power forecasting method based on LSTM-RNN model and time correlation modification under partial daily pattern prediction framework”. In: *Energy Conversion and Management* 212 (2020), p. 112766.
- [27] Gangqiang Li et al. “Photovoltaic power forecasting with a hybrid deep learning approach”. In: *IEEE Access* 8 (2020), pp. 175871–175880.
- [28] Elizaveta Kharlova, Daniel May, and Petr Musilek. “Forecasting photovoltaic power production using a deep learning sequence to sequence model with attention”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–7.
- [29] Yusen Wang, Wenlong Liao, and Yuqing Chang. “Gated Recurrent Unit Network-Based Short-Term Photovoltaic Forecasting”. In: *Energies* 11.8 (2018). ISSN: 1996-1073. DOI: 10.3390/en11082163. URL: <https://www.mdpi.com/1996-1073/11/8/2163>.

- [30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [31] Maryam Hashemi, Alireza Mirrashid, and Aliasghar Beheshti Shirazi. “Driver safety development: Real-time driver drowsiness detection system based on convolutional neural network”. In: *SN Computer Science* 1.5 (2020), pp. 1–10.
- [32] Mohit Dua et al. “Deep CNN models-based ensemble approach to driver drowsiness detection”. In: *Neural Computing and Applications* 33.8 (2021), pp. 3155–3168.
- [33] Jing-Ming Guo and Herleeyandi Markoni. “Driver drowsiness detection using hybrid convolutional neural network and long short-term memory”. In: *Multimedia tools and applications* 78.20 (2019), pp. 29059–29087.
- [34] Prima Dewi Purnamasari and Aziz Zul Hazmi. “Heart beat based drowsiness detection system for driver”. In: *2018 International Seminar on Application for Technology of Information and Communication*. IEEE. 2018, pp. 585–590.
- [35] Lee Boon Leng, Lee Boon Giin, and Wan-Young Chung. “Wearable driver drowsiness detection system based on biomedical and motion sensors”. In: *2015 IEEE SENSORS*. IEEE. 2015, pp. 1–4.
- [36] Miriam R Waldeck and Michael I Lambert. “Heart rate during sleep: implications for monitoring training status”. In: *Journal of sports science & medicine* 2.4 (2003), p. 133.
- [37] Difei Jing, Shuwei Zhang, and Zhongyin Guo. “Fatigue driving detection method for low-voltage and Hypoxia Plateau Area: a physiological character-

- istic analysis approach”. In: *International journal of transportation science and technology* 9.2 (2020), pp. 148–158.
- [38] Xuemin Zhu et al. “EOG-based drowsiness detection using convolutional neural networks”. In: *2014 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2014, pp. 128–134.
- [39] Günther Deuschl. “Recommendations for the practice of clinical neurophysiology”. In: *guidelines of the International Federation of Clinical Neurophysiology* (1999).
- [40] Hong J Eoh, Min K Chung, and Seong-Han Kim. “Electroencephalographic study of drowsiness in simulated driving with sleep deprivation”. In: *International Journal of Industrial Ergonomics* 35.4 (2005), pp. 307–320.
- [41] Claudio A Perez et al. “Face and eye tracking algorithm based on digital image processing”. In: *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat. No. 01CH37236)*. Vol. 2. IEEE. 2001, pp. 1178–1183.
- [42] Sarbjit Singh and Nikolaos P Papanikolopoulos. “Monitoring driver fatigue using facial analysis techniques”. In: *Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat. No. 99TH8383)*. IEEE. 1999, pp. 314–318.
- [43] Konstantinos P Moustris, Ioannis C Ziomas, and Athanasios G Paliatsos. “3-Day-ahead forecasting of regional pollution index for the pollutants NO₂, CO, SO₂, and O₃ using artificial neural networks in Athens, Greece”. In: *Water, Air, & Soil Pollution* 209.1 (2010), pp. 29–43.

- [44] Fabio Biancofiore et al. “Recursive neural network model for analysis and forecast of PM10 and PM2.5”. In: *Atmospheric Pollution Research* 8.4 (2017), pp. 652–659.
- [45] Sheen Mclean S Cabaneros, John Kaiser S Calautit, and Ben Richard Hughes. “Hybrid artificial neural network models for effective prediction and mitigation of urban roadside NO2 pollution”. In: *Energy Procedia* 142 (2017), pp. 3524–3530.
- [46] Samuel D Lightstone, Fred Moshary, and Barry Gross. “Comparing CMAQ forecasts with a neural network forecast model for PM2.5 in New York”. In: *Atmosphere* 8.9 (2017), p. 161.
- [47] Ming-Tung Chuang, Yang Zhang, and Daiwen Kang. “Application of WRF/Chem-MADRID for real-time air quality forecasting over the Southeastern United States”. In: *Atmospheric environment* 45.34 (2011), pp. 6241–6250.
- [48] Stephen F Mueller and Jonathan W Mallard. “Contributions of natural emissions to ozone and PM2.5 as simulated by the community multiscale air quality (CMAQ) model”. In: *Environmental science & technology* 45.11 (2011), pp. 4817–4823.
- [49] Sheen Mclean Cabaneros, John Kaiser Calautit, and Ben Richard Hughes. “A review of artificial neural network models for ambient air pollution prediction”. In: *Environmental Modelling & Software* 119 (2019), pp. 285–304.
- [50] Xiao Feng et al. “Artificial neural networks forecasting of PM2.5 pollution using air mass trajectory based geographic model and wavelet transformation”. In: *Atmospheric Environment* 107 (2015), pp. 118–128.

- [51] Fang Zhao and Weide Li. “A combined model based on feature selection and WOA for PM 2.5 concentration forecasting”. In: *Atmosphere* 10.4 (2019), p. 223.
- [52] Yue-Shan Chang et al. “An LSTM-based aggregated model for air pollution forecasting”. In: *Atmospheric Pollution Research* 11.8 (2020), pp. 1451–1463.
- [53] Jingyang Wang et al. “Air quality prediction using CT-LSTM”. In: *Neural Computing and Applications* 33.10 (2021), pp. 4779–4792.
- [54] Kunwar P Singh et al. “Linear and nonlinear modeling approaches for urban air quality prediction”. In: *Science of the Total Environment* 426 (2012), pp. 244–255.
- [55] Wenjian Wang, Changqian Men, and Weizhen Lu. “Online prediction model based on support vector machine”. In: *Neurocomputing* 71.4-6 (2008), pp. 550–558.
- [56] Victor R Prybutok, Junsob Yi, and David Mitchell. “Comparison of neural network models with ARIMA and regression models for prediction of Houston’s daily maximum ozone concentrations”. In: *European Journal of Operational Research* 122.1 (2000), pp. 31–40.
- [57] Lele Qin, Naiwen Yu, and Donghui Zhao. “Applying the convolutional neural network deep learning technology to behavioural recognition in intelligent video”. In: *Tehnički vjesnik* 25.2 (2018), pp. 528–535.
- [58] Fatih Taşpınar. “Improving artificial neural network model predictions of daily average PM10 concentrations by applying principle component analysis and implementing seasonal models”. In: *Journal of the Air & Waste Management Association* 65.7 (2015), pp. 800–809.

- [59] Bun Theang Ong, Komei Sugiura, and Koji Zettsu. “Dynamically pre-trained deep recurrent neural networks using environmental monitoring data for predicting PM_{2.5}”. In: *Neural Computing and Applications* 27.6 (2016), pp. 1553–1566.
- [60] Esteban Pardo and Norberto Malpica. “Air quality forecasting in Madrid using long short-term memory networks”. In: *International Work-Conference on the Interplay Between Natural and Artificial Computation*. Springer. 2017, pp. 232–239.
- [61] Xianghong Wang and Baozhen Wang. “Research on prediction of environmental aerosol and PM_{2.5} based on artificial neural network”. In: *Neural Computing and Applications* 31.12 (2019), pp. 8217–8227.
- [62] Ebrahim Eslami et al. “A real-time hourly ozone prediction system using deep convolutional neural network”. In: *Neural Computing and Applications* 32.13 (2020), pp. 8783–8797.
- [63] Kuiying Gu et al. “Prediction of air quality in Shenzhen based on neural network algorithm”. In: *Neural Computing and Applications* 32.7 (2020), pp. 1879–1892.
- [64] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P Mathur. “WADI: a water distribution testbed for research in the design of secure cyber physical systems”. In: *Proceedings of the 3rd international workshop on cyber-physical systems for smart water networks*. 2017, pp. 25–28.
- [65] Sawsan Khaleel Alshattnawi. “Smart water distribution management system architecture based on internet of things and cloud computing”. In: *2017 International Conference on New Trends in Computing Sciences (ICTCS)*. IEEE. 2017, pp. 289–294.

- [66] Lakshmi Kanthan Narayanan and Suresh Sankaranarayanan. “IoT-based water demand forecasting and distribution design for smart city”. In: *Journal of Water and Climate Change* 11.4 (2020), pp. 1411–1428.
- [67] SV Mohanasundaram et al. “Smart water distribution network solution for smart cities: Indian scenario”. In: *2018 Global Internet of Things Summit (GIoTS)*. IEEE. 2018, pp. 1–6.
- [68] Michele Romano and Zoran Kapelan. “Adaptive water demand forecasting for near real-time management of smart water distribution systems”. In: *Environmental Modelling & Software* 60 (2014), pp. 265–276.
- [69] Alexandru Predescu et al. “An advanced learning-based multiple model control supervisor for pumping stations in a smart water distribution system”. In: *Mathematics* 8.6 (2020), p. 887.
- [70] Wanqing Zhao, Thomas H Beach, and Yacine Rezgui. “Optimization of potable water distribution and wastewater collection networks: A systematic review and future research directions”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 46.5 (2015), pp. 659–681.
- [71] Tomas Robles et al. “An IoT based reference architecture for smart water management processes.” In: *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.* 6.1 (2015), pp. 4–23.
- [72] Mohamed Afifi, Mohamed F Abdelkader, and Atef Ghoneim. “An IoT system for continuous monitoring and burst detection in intermittent water distribution networks”. In: *2018 International Conference on Innovative Trends in Computer Engineering (ITCE)*. IEEE. 2018, pp. 240–247.

- [73] Rosiberto Gonçalves, Jesse JM Soares, and Ricardo MF Lima. “An IoT-based framework for smart water supply systems management”. In: *Future Internet* 12.7 (2020), p. 114.
- [74] Gabriela Cembrano et al. “Optimal control of a water distribution network in a supervisory control system”. In: *Control engineering practice* 8.10 (2000), pp. 1177–1188.
- [75] E Creaco et al. “Real time control of water distribution networks: A state-of-the-art review”. In: *Water research* 161 (2019), pp. 517–530.
- [76] Ilyas Eker and Tolgay Kara. “Operation and control of a water supply system”. In: *ISA transactions* 42.3 (2003), pp. 461–473.
- [77] Paul W Jowitt and Chengchao Xu. “Optimal valve control in water-distribution networks”. In: *Journal of Water Resources Planning and Management* 116.4 (1990), pp. 455–472.
- [78] Nehemiah Musa et al. “A systematic review and Meta-data analysis on the applications of Deep Learning in Electrocardiogram”. In: *Journal of ambient intelligence and humanized computing* (2022), pp. 1–74.
- [79] Adel Mellit et al. “Advanced methods for photovoltaic output power forecasting: A review”. In: *Applied Sciences* 10.2 (2020), p. 487.
- [80] Mei Yang et al. “Deep learning algorithms and multicriteria decision-making used in big data: a systematic literature review”. In: *Complexity* 2020 (2020).
- [81] Nhi NY Vo et al. “Deep learning for decision making and the optimization of socially responsible investments and portfolio”. In: *Decision Support Systems* 124 (2019), p. 113097.

- [82] Iqbal H Sarker. “Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions”. In: *SN Computer Science* 2.6 (2021), pp. 1–20.
- [83] Rene Y Choi et al. “Introduction to machine learning, neural networks, and deep learning”. In: *Translational Vision Science & Technology* 9.2 (2020), pp. 14–14.
- [84] Vetle Øyri. “Long Short-term Memory (LSTM) recurrent neural networks for urban hydrological modelling”. MA thesis. 2020.
- [85] Jafar Alzubi, Anand Nayyar, and Akshi Kumar. “Machine learning from theory to algorithms: an overview”. In: *Journal of physics: conference series*. Vol. 1142. 1. IOP Publishing. 2018, p. 012012.
- [86] Dmytro Mishkin, Nikolay Sergievskiy, and Jiri Matas. “Systematic evaluation of convolution neural network advances on the imagenet”. In: *Computer vision and image understanding* 161 (2017), pp. 11–19.
- [87] Gopinath Rebala, Ajay Ravi, and Sanjay Churiwala. *An introduction to machine learning*. Springer, 2019.
- [88] Oleksii Trekhleb. *machine-learning-map.png*. <https://github.com/trekhleb/homemade-machine-learning/blob/master/images/machine-learning-map.png>. [Online; accessed 12-October-2022]. 2018.
- [89] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.
- [90] Rocio Vargas, Amir Mosavi, and Ramon Ruiz. “Deep learning: a review”. In: (2017).

- [91] Rong Zhang, Weiping Li, and Tong Mo. “Review of deep learning”. In: *arXiv preprint arXiv:1804.01653* (2018).
- [92] Laith Alzubaidi et al. “Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions”. In: *Journal of big Data* 8.1 (2021), pp. 1–74.
- [93] Maryam M Najafabadi et al. “Deep learning applications and challenges in big data analytics”. In: *Journal of big data* 2.1 (2015), pp. 1–21.
- [94] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [95] Salman Khan et al. “A guide to convolutional neural networks for computer vision”. In: *Synthesis lectures on computer vision* 8.1 (2018), pp. 1–207.
- [96] Weili Fang et al. “Computer vision for behaviour-based safety in construction: A review and future directions”. In: *Advanced Engineering Informatics* 43 (2020), p. 100980.
- [97] Ruoyu Yang et al. “CNN-LSTM deep learning architecture for computer vision-based modal frequency detection”. In: *Mechanical Systems and signal processing* 144 (2020), p. 106885.
- [98] Ashutosh Pandey and DeLiang Wang. “A new framework for CNN-based speech enhancement in the time domain”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27.7 (2019), pp. 1179–1188.
- [99] Dimitri Palaz, Mathew Magimai-Doss, and Ronan Collobert. “End-to-end acoustic modeling using convolutional neural networks for HMM-based au-

- automatic speech recognition”. In: *Speech Communication* 108 (2019), pp. 15–32.
- [100] Dimitri Palaz, Ronan Collobert, et al. *Analysis of CNN-based speech recognition system using raw speech as input*. Tech. rep. Idiap, 2015.
- [101] Samil Karahan et al. “How image degradations affect deep CNN-based face recognition?” In: *2016 international conference of the biometrics special interest group (BIOSIG)*. IEEE. 2016, pp. 1–5.
- [102] Hsiao-Chi Li, Zong-Yue Deng, and Hsin-Han Chiang. “Lightweight and resource-constrained learning network for face recognition with performance optimization”. In: *Sensors* 20.21 (2020), p. 6114.
- [103] Jie Wang and Zihao Li. “Research on face recognition based on CNN”. In: *IOP Conference Series: Earth and Environmental Science*. Vol. 170. 3. IOP Publishing. 2018, p. 032110.
- [104] Ajitesh Kumar. *Different Types of CNN Architectures Explained: Examples*. <https://vitalflux.com/wp-content/uploads/2021/11/VGG16-CNN-Architecture.png>. [Online; accessed 19-October-2022]. 2022.
- [105] Etienne Dupuis et al. “CNN weight sharing based on a fast accuracy estimation metric”. In: *Microelectronics Reliability* 122 (2021), p. 114148.
- [106] Peng Wang, Xiaomin Zhang, and Yan Hao. “A method combining CNN and ELM for feature extraction and classification of SAR image”. In: *Journal of Sensors* 2019 (2019).
- [107] Mengmeng Zhang et al. “Feature extraction for classification of hyperspectral and LiDAR data using patch-to-patch CNN”. In: *IEEE transactions on cybernetics* 50.1 (2018), pp. 100–111.

- [108] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. “Understanding of a convolutional neural network”. In: *2017 international conference on engineering and technology (ICET)*. Ieee. 2017, pp. 1–6.
- [109] Keiron O’Shea and Ryan Nash. “An introduction to convolutional neural networks”. In: *arXiv preprint arXiv:1511.08458* (2015).
- [110] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. “Activation functions in neural networks”. In: *towards data science* 6.12 (2017), pp. 310–316.
- [111] Forest Agostinelli et al. “Learning activation functions to improve deep neural networks”. In: *arXiv preprint arXiv:1412.6830* (2014).
- [112] Pragati Baheti. *Activation Functions in Neural Networks [12 Types & Use Cases]*. <https://www.v7labs.com/blog/neural-networks-activation-functions>. [Online; accessed 14-October-2022]. 2022.
- [113] NBSHARE NOTEBOOKS. *Activation Functions In Python*. <https://www.nbshare.io/notebook/751082217/Activation-Functions-In-Python/>. [Online; accessed 16-October-2022]. 2022.
- [114] Javier Antonanzas et al. “Review of photovoltaic power forecasting”. In: *Solar Energy* 136 (2016), pp. 78–111.
- [115] Nagesh Singh Chauhan. *Loss Functions in Neural Networks*. <https://www.theaidream.com/post/loss-functions-in-neural-networks>. [Online; accessed 12-October-2022]. 2021.
- [116] Prashanth Saravanan. *Understanding Loss Functions in Machine Learning*. <https://www.section.io/engineering-education/understanding-loss-functions-in-machine-learning/>. [Online; accessed 16-October-2022]. 2022.

- [117] C3.ai. *Root Mean Square Error (RMSE)*. <https://c3.ai/glossary/data-science/root-mean-square-error-rmse/#:~:text=What%20is%20Root%20Mean%20Square,true%20values%20using%20Euclidean%20distance..> [Online; accessed 17-October-2022]. 2022.
- [118] Shubham Jain. *An Overview of Regularization Techniques in Deep Learning (with Python code)*. <https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/>. [Online; accessed 12-October-2022]. 2018.
- [119] Federico Girosi, Michael Jones, and Tomaso Poggio. “Regularization theory and neural networks architectures”. In: *Neural computation* 7.2 (1995), pp. 219–269.
- [120] Editorial Team. *Improving Artificial Neural Network with Regularization and Optimization*. <https://towardsai.net/p/machine-learning/improving-artificial-neural-network-with-regularization-and-optimization>. [Online; accessed 12-October-2022]. 2020.
- [121] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.
- [122] Muhammad Yaqub et al. “State-of-the-art CNN optimizer for brain tumor segmentation in magnetic resonance images”. In: *Brain Sciences* 10.7 (2020), p. 427.
- [123] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016).

- [124] Léon Bottou. “Large-scale machine learning with stochastic gradient descent”. In: *Proceedings of COMPSTAT’2010*. Springer, 2010, pp. 177–186.
- [125] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. “Neural networks for machine learning lecture 6a overview of mini-batch gradient descent”. In: *Cited on 14.8* (2012), p. 2.
- [126] Zijun Zhang. “Improved adam optimizer for deep neural networks”. In: *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. Ieee. 2018, pp. 1–2.
- [127] Ange Tato and Roger Nkambou. “Improving adam optimizer”. In: (2018).
- [128] Laith Alzubaidi et al. “Optimizing the performance of breast cancer classification by employing the same domain transfer learning from hybrid deep convolutional neural network model”. In: *Electronics* 9.3 (2020), p. 445.
- [129] Evangelos Bousias Alexakis and Costas Armenakis. “Performance Improvement of Encoder/Decoder-Based CNN Architectures for Change Detection from Very High-Resolution Satellite Imagery”. In: *Canadian Journal of Remote Sensing* 47.2 (2021), pp. 309–336.
- [130] Parul Sharma, Yash Paul Singh Berwal, and Wiqas Ghai. “Performance analysis of deep learning CNN models for disease detection in plants using image segmentation”. In: *Information Processing in Agriculture* 7.4 (2020), pp. 566–574.
- [131] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

- [132] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.
- [133] Alex Graves et al. “A novel connectionist system for unconstrained handwriting recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 31.5 (2008), pp. 855–868.
- [134] Victor Carbune et al. “Fast multi-language LSTM-based online handwriting recognition”. In: *International Journal on Document Analysis and Recognition (IJ DAR)* 23.2 (2020), pp. 89–102.
- [135] Curtis Wigington et al. “Data augmentation for recognition of handwritten words and lines using a CNN-LSTM network”. In: *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*. Vol. 1. IEEE. 2017, pp. 639–645.
- [136] Weicong Kong et al. “Short-term residential load forecasting based on LSTM recurrent neural network”. In: *IEEE Transactions on Smart Grid* 10.1 (2017), pp. 841–851.
- [137] Huiting Zheng, Jiabin Yuan, and Long Chen. “Short-term load forecasting using EMD-LSTM neural networks with a Xgboost algorithm for feature importance evaluation”. In: *Energies* 10.8 (2017), p. 1168.
- [138] Bo-Sung Kwon, Rae-Jun Park, and Kyung-Bin Song. “Short-term load forecasting based on deep neural networks using LSTM layer”. In: *Journal of Electrical Engineering & Technology* 15.4 (2020), pp. 1501–1509.
- [139] Zheng Zhao et al. “LSTM network: a deep learning approach for short-term traffic forecast”. In: *IET Intelligent Transport Systems* 11.2 (2017), pp. 68–75.

- [140] Toon Bogaerts et al. “A graph CNN-LSTM neural network for short and long-term traffic forecasting based on trajectory data”. In: *Transportation Research Part C: Emerging Technologies* 112 (2020), pp. 62–77.
- [141] Rusul L Abduljabbar et al. “Short-term traffic forecasting: an LSTM network for spatial-temporal speed prediction”. In: *Future Transportation* 1.1 (2021), pp. 21–37.
- [142] Yong Yu et al. “A review of recurrent neural networks: LSTM cells and network architectures”. In: *Neural computation* 31.7 (2019), pp. 1235–1270.
- [143] Klaus Greff et al. “LSTM: A search space odyssey”. In: *IEEE transactions on neural networks and learning systems* 28.10 (2016), pp. 2222–2232.
- [144] Christopher Olah. *Understanding LSTM Networks*. <https://colah.github.io/posts/2015-understanding-lstm-1/>. [Online; accessed 17-October-2022]. 2015.
- [145] Mostafa Zaman et al. “A Deep Learning Model for Forecasting Photovoltaic Energy with Uncertainties”. In: *2021 IEEE Green Energy and Smart Systems Conference (IGESSC)*. IEEE. 2021, pp. 1–6.
- [146] P Viola and M Jones. “Face detection”. In: *IJCV* 57 (2004), p. 2.
- [147] Kevan Yuen and Mohan M Trivedi. “An occluded stacked hourglass approach to facial landmark localization and occlusion estimation”. In: *IEEE Transactions on Intelligent Vehicles* 2.4 (2017), pp. 321–331.
- [148] Mostafa Zaman, Nasibeh Zohrabi, and Sherif Abdelwahed. “A CNN-based Path Trajectory Prediction Approach with Safety Constraints”. In: *2020 IEEE*

- Transportation Electrification Conference & Expo (ITEC)*. IEEE. 2020, pp. 267–272.
- [149] Yan Zhang and Caijian Hua. “Driver fatigue recognition based on facial expression analysis using local binary patterns”. In: *Optik* 126.23 (2015), pp. 4501–4505.
- [150] Jun-Juh Yan et al. “Real-time driver drowsiness detection system based on PERCLOS and grayscale image processing”. In: *2016 International Symposium on Computer, Consumer and Control (IS3C)*. IEEE. 2016, pp. 243–246.
- [151] USEPA_2012. *Technical assistance document for the reporting of Daily Air Quality*. <https://www.airnow.gov/publications/air-quality-index/technical-assistance-document-for-reporting-the-daily-aqi/>. [Online; accessed 20-September-2022]. 2018.
- [152] Anikender Kumar and Piyush Goyal. “Forecasting of air quality index in Delhi using neural network based on principal component analysis”. In: *Pure and Applied Geophysics* 170.4 (2013), pp. 711–722.
- [153] Jose Antonio Moscoso-López et al. “Hourly air quality index (AQI) forecasting using machine learning methods”. In: *International Workshop on Soft Computing Models in Industrial and Environmental Applications*. Springer. 2020, pp. 123–132.
- [154] USEPA. *Air Data: Air Quality Data Collected at Outdoor Monitors Across the US*. <https://www.epa.gov/outdoor-air-quality-data>. [Online; accessed 19-September-2022]. 2018.

- [155] Eng Hwee Ong et al. “IEEE 802.11 ac: Enhancements for very high throughput WLANs”. In: *2011 IEEE 22nd International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE. 2011, pp. 849–853.
- [156] RA Atmoko, R Riantini, and MK Hasin. “IoT real time data acquisition using MQTT protocol”. In: *Journal of Physics: Conference Series*. Vol. 853. 1. IOP Publishing. 2017, p. 012003.
- [157] Michael Buckland and Fredric Gey. “The relationship between recall and precision”. In: *Journal of the American society for information science* 45.1 (1994), pp. 12–19.
- [158] Jiaju Miao and Wei Zhu. “Precision–recall curve (PRC) classification trees”. In: *Evolutionary intelligence* 15.3 (2022), pp. 1545–1569.
- [159] Sofia Visa et al. “Confusion matrix-based feature selection”. In: *MAICS* 710.1 (2011), pp. 120–127.
- [160] Y Arafah and H Winarso. “Redefining smart city concept with resilience approach”. In: *IOP conference series: earth and environmental science*. Vol. 70. 1. IOP Publishing. 2017, p. 012065.
- [161] Mircea Eremia, Lucian Toma, and Mihai Sanduleac. “The smart city concept in the 21st century”. In: *Procedia Engineering* 181 (2017), pp. 12–19.
- [162] Irina Picioroaga et al. “Resilient operation of distributed resources and electrical networks in a smart city context”. In: *UPB Sci. Bull., Series C* 82.3 (2020), pp. 267–278.

VITA

Mostafa Zaman's graduate work has focused mainly on incorporating a machine learning method to deal with uncertainty in various areas of a smart city. He has also aided in teaching undergraduate and graduate students in his graduate studies, such as digital logic design, industrial automation, C and C++ programming, digital systems, introduction to the cyber-physical system, etc. His principal research focuses on building an uncertainty-aware decision support system in different aspects of smart cities and analyzing analytical data applications using python. He also focused on constructing Smart Building and Smart Water distribution systems on the OpenCyberCity testbed and performing data analytics.