REAL TIME AGRICULTURE SURVEILLANCE USING DEEP LEARNING

A PROJECT REPORT

Submitted by

PRAVIEEN AS [Reg No:RA2011003011015]

RAJAGOPAL C [Reg No:RA2011003010994]

Under the Guidance of

Dr. T. MANORANJITHAM

Associate Professor, Department of Computing Technologies

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGIES

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR– 603 203

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR–603 203

BONAFIDE CERTIFICATE

Certified that 18CSP107L / 18CSP108L - MAJOR PROJECT report titled **"Real Time Agriculture Surveillance using Deep Learning"** is the bonafide work of **PRAVIEEN AS [Reg No: RA2011003011015]** and **RAJAGOPAL C [Reg No: RA201103010994]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge, the work reported here does not form part of any other thesis or dissertation based on which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr. T. Manoranjitham
SUPERVISOR
Associate Professor
Department of Computing Technologies

Dr. T. Manoranjitham
PANEL HEAD
Associate Professor
Department of Computing Technologies

Dr. M. PUSHPALATHA
HEAD OF THE DEPARTMENT
Department of Computing Technologies

INTERNAL EXAMINER

EXTERNAL EXAMINER

Department of Computing Technologies
SRM Institute of Science and Technology
Own Work Declaration Form

Degree/Course: B.Tech in Computer Science and Engineering

Student Names: PRAVIEEN AS, RAJAGOPAL C

Registration Number: RA201103011015, RA2011003010994

Title of Work: **REAL-TIME AGRICULTURE SURVEILLANCE USING DEEP LEARNING**
I/We here by certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.
I / We confirm that all the work contained in this assessment is our own except where indicated and that we have met the following conditions:

- References / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data, etc that are not my own.
- Not making any use of the report(s) or essay(s) of any other student(s)either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

I understand that any false claim for this work will be penalized by the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our work, except where indicated by referring, and that I have followed the good academic practices noted above.
Pravieen AS Signature:

Rajagopal C Signature:

Date:25.04.2024

If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ACKNOWLEDGEMENT

We express our humble gratitude to **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology,
Dr. T. V. Gopal, for his invaluable support.

We wish to thank **Dr. Revathi Venkataraman**, Professor and Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, **Dr. M. Pushpalatha,** Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for her suggestions and encouragement at all stages of the project work.

We want to convey our thanks to our Project Coordinators, **S. Godfrey Winster,** Associate Professor,
Dr. M. Baskar, Associate Professor, **Dr. P. Murali,** Associate Professor, **Dr. J. Selvin Paul Peter,** Associate Professor, **Dr. C. Pretty Diana Cyril** Assistant Professor **and Dr. G. Padmapriya**, Panel Head,
Dr.T. Manoranjitham, Associate Professor and Panel Members, **Dr. Vidhya S**, Assistant Professor**,**
Dr. Kalaivani J, Associate Professor**,** and **Dr. Priya Milton**, Assistant Professor Department of Computing Technologies, SRM Institute of Science and Technology, for their inputs during the project reviews and support.

We register our thanks to our Faculty Advisors, **Dr. Robert P**, Assistant Professor, and

Dr. AMJ Muthukumaran, Assistant Professor Department of Computing Technologies, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr. T. Manoranjitham**, Associate Professor, Department of Computing Technologies, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under her mentorship. She provided us with the freedom and support to explore the research topics of our interest. Her passion for solving problems and making a difference in the world has always been inspiring.

We sincerely thank all the staff and students of the Computing Technologies Department, School of Computing, S.R.M Institute of Science and Technology, for their help during our project. Finally, we would like to thank our parents, family members, and friends for their unconditional love, constant support, and encouragement.

Pravieen AS [RegNo: RA2011003011015]

Rajagopal C [RegNo: RA2011003010994]

ABSTRACT

In the pursuit of enhancing agricultural sustainability and crop protection through technological innovation, our research introduces a groundbreaking application of Convolutional Neural Networks (CNN) via an auto-encoder model for the detection of non-native animal intrusions in agricultural settings. This initiative is an urgent need to safeguard crops and maintain farming efficiency by identifying and mitigating the presence of animals that are typically alien to such environments. The auto-encoder, a sophisticated neural network model tailored for unsupervised learning, distills images into a condensed form. This process facilitates the effective identification of anomalies by spotlighting discrepancies through reconstruction errors, thereby enabling the precise detection of atypical animal activity within farm imagery. Central to our methodology is a meticulously curated dataset, which has been significantly informed by the Animal-10 dataset available on Kaggle. This Animal-10 dataset is carefully segmented into subfiles; one represents the normative scenario, featuring images of animals usually found on farms, while the other catalogs anomalies, encompassing unexpected visitors such as bears and lions. This comprehensive compilation, boasting over 3,000 images, serves as a solid foundation for both training and validating our model. The diversity and specificity of the dataset ensure a robust testing ground for the auto-encoder model, enhancing its ability to generalize and accurately identify anomalies in a real-world agricultural context. The model's performance, with an accuracy exceeding 90%, underscores its efficacy as a potent tool for farmers and agricultural stakeholders, offering a novel approach to pre-emptively address potential threats to crop health and farm productivity.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
KNN	K-Nearest Neighbors
SVM	Support Vector Machine
ANN	Artificial Neural Networks

CHAPTER 1

INTRODUCTION

Using cutting-edge technologies into agricultural methods has become a key tactic in tackling the complex issues of crop protection, sustainability, and productivity improvement. The use of deep learning and machine learning techniques stands out among these technologies because it has the potential to completely transform conventional farming practices. This paper delves into the innovative use of Convolutional Neural Networks (CNN) through an auto-encoder model, aimed at detecting the presence of non-native animals in agricultural settings—an issue of significant concern for maintaining crop health and farm efficiency. The necessity to develop such a system arises from the increasing instances of wildlife encroachments into agricultural lands, which not only threaten crop yield but also pose challenges to wildlife management and farm safety.

Building on the foundation of a custom-built dataset, informed and expanded upon the basis of the Animal-10 dataset from Kaggle, our research outlines the creation and utilization of a dataset comprising over 3,000 images. This dataset is meticulously categorized into normal instances (depicting animals typically found in farm settings) and anomalies (showcasing non-native animals such as bears and lions), providing a comprehensive basis for training and evaluating our proposed model. The auto-encoder model, known for its efficiency in unsupervised learning through the process of dimensionality reduction and anomaly detection via reconstruction error, has been benchmarked against traditional and contemporary models, namely K-Nearest Neighbors (KNN) and Isolation Forest, across various parameters.

The comparative analysis focused on several critical aspects: the level of supervision required, the type of learning (supervised, unsupervised, or semi-supervised), the nature of the data handled, sensitivity to outliers, dimensionality, robustness to noise, training time, and versatility of application. This extensive evaluation reveals that the auto-encoder model not only excels in detecting anomalies within the agricultural context but also offers substantial advantages in terms of noise robustness, handling high-dimensional data, and efficiency in training time, making it a superior choice among the models tested.

In addition to offering the auto-encoder model as a workable approach to anomaly detection in agriculture, the purpose of this study is to further the conversation on the potential uses of deep learning technology to improve agricultural practices. This research sheds light on the essential elements that determine the efficacy of anomaly detection systems in agricultural settings by comparing various models under a single set of parameters, opening the door for future advancements in smart farming technology.

1.1 Machine Learning

Machine learning is the branch of artificial intelligence that deals with building models and algorithms that can learn from data to produce judgments and predictions. With reference to the models under discussion, machine learning demonstrates its fundamental principles.

Machine learning techniques, such as Decision Trees, Support Vector Machines, Logistic Regression, Random Forests, K-nearest neighbors, Naive Bayes, Neural Networks, and XGBoost, are used to analyze and process data. Their objective is to mine databases for links, patterns, and insights that will enable automated decision-making.

In order to make predictions and draw generalizations about fresh, unknown data, these models are trained using prior data. With training and fine-tuning, machine learning models modify their internal parameters to optimize their performance with the goal of achieving greater accuracy and reliability. Beyond agricultural applications, machine learning is widely used in recommendation systems, driverless vehicles, picture recognition, and natural language processing. It is a powerful approach to data analysis and problem solving that empowers systems to make informed decisions and provide insightful information across a range of domains. Machine learning is essentially the science of teaching computers to learn from data, which helps them to grow more intelligent and skilled at executing complex jobs.

.

1.2 Deep Learning

Deep learning is a subfield of machine learning that specializes in multi-layered neural networks, or deep neural networks. The structure and operations of the human brain served as inspiration for these networks, which analyze data through interconnected layers of artificial neurons.
When it comes to automatically recognizing and expressing complex characteristics and patterns in data, deep learning excels.

Three layers comprise deep neural networks: an input layer, an output layer, and hidden layers that perform tasks like feature extraction and abstraction.
One benefit of deep learning is that it can automatically learn features, eliminating the need for human feature engineering.
Large datasets are often required for deep learning models to be trained in order for them to generate accurate predictions and become more general.
Applications for deep learning can be found in a wide range of industries, including speech and picture identification, autonomous cars, natural language processing, and healthcare.
Deep learning models outperform conventional machine learning methods when it comes to intricate patterns and big datasets.
This is a significant advancement in artificial intelligence that enables machines to learn, adapt, and make decisions just like humans.

1.3 Difference Between Machine Learning And Deep Learning

Machine learning involves algorithms that allow computers to learn from data and make predictions, while deep learning is a subset focusing on neural networks, especially deep ones with multiple layers, enabling complex data representation learning.

ATTRIBUTE	DEEP LEARNING	MACHINE LEARNING
Architecture and Model Complexity	Deep learning employs multiple-layer neural networks, making it appropriate for complicated tasks. These networks learn hierarchical representations on their own**.**	Traditional machine learning models typically contain fewer layers and are shallower in general, necessitating feature engineering to extract useful information.
Feature Extraction	Deep learning is particularly good at automatic feature learning, which eliminates the need for manual feature engineering. It can learn relevant features directly from raw data**.**	Feature engineering is a critical phase in classical ML, where domain knowledge is used to extract and choose relevant features for the model.

Data Size	Deep learning frequently necessitates big datasets to operate well. The large amount of data allows it to generalize and generate reliable forecasts.	Traditional machine learning can be effective with smaller datasets and does not always require massive volumes of data.
Interpretability	Deep learning models might be challenging to understand Because of the complexity of the network structures, it may be difficult to explain why a given prediction was made.	Traditional machine learning models are frequently more interpretable. Decision trees, for example, provide a clear understanding of how predictions are made.
Computational Resources	Deep learning models are computationally demanding and frequently necessitate the use of specialized hardware(e.g., GPUs) for training.	Traditional machine learning models are less computationally intensive and can be trained on regular hardware.
Versatility	Deep learning is ideal for image and speech identification, natural language processing, and sophisticated pattern recognition.	Traditional machine learning is versatile, addressing a wide range of tasks like recommendation, clustering, regression, and classification
Accuracy vs. Simplicity	Deep learning models produce high accuracy but can be too complex for minor tasks.	Traditional ML models are simpler and may be enough for many applications requiring modest accuracy.
Training Time	Because of their complexity and data amount, deep learning models frequently necessitate longer training cycles.	Training times for traditional ML models are often shorter.

1.4 How It Will Help In Anomaly Detection ?

With multiple benefits over conventional techniques, deep learning has become a potent instrument for anomaly detection in a variety of fields, including industrial systems, finance, cybersecurity, and healthcare. Deep learning's capacity to automatically discover intricate patterns and representations from unprocessed data, without the need for hand-crafted features or domain-specific expertise, is one of its fundamental advantages. This is especially helpful for tasks involving anomaly identification, as anomalies sometimes display complex and nuanced patterns that are challenging to identify with conventional feature engineering techniques. Deep learning models are capable of learning hierarchical representations of data at multiple levels of abstraction. This allows them to identify anomalies in a variety of data types, including text, images, time-series, and sensor data. Examples of these models are deep neural networks (DNNs),
convolutional neural networks (CNNs), recurrent neural networks (RNNs), and their variations.

Deep learning's capacity to handle high-dimensional data with intricate connections and non-linear correlations is one of its main advantages in anomaly identification. Such data are often difficult to model using traditional statistical techniques, particularly when working with large-scale datasets or high-velocity, high-volume data streams. In contrast, deep learning models provide intrinsic flexibility and scalability, enabling them to identify complex patterns and grasp the fundamental structure of the data, even when noise and unpredictability are present. This makes it possible for deep learning-based anomaly detection systems to outperform conventional methods in terms of accuracy and robustness, particularly in real-world situations where data may be extremely dynamic and varied.Additionally, deep learning has the benefit of end-to-end learning, which allows for the unified, data-driven training of the complete anomaly detection pipeline, optimizing both anomaly detection and feature extraction at the same time.

This is in contrast to conventional techniques, which frequently entail an algorithm for anomaly identification after a pipeline of manually constructed feature extraction. Deep learning models are able to automatically adjust to the properties of the data by utilizing end-to-end learning. This allows the models to learn discriminative features straight from the raw input and successfully distinguish between normal and abnormal patterns. This improves the anomaly detection system's overall performance and capacity for generalization in addition to streamlining the design process.

Deep learning's proficiency with handling sequential and temporal data is another asset in anomaly detection. Time-series data analysis is used in many real-world applications, including cybersecurity, network intrusion detection, and predictive maintenance, to find unusual behavior or events. The specialized architectures for processing sequential data, recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) networks, have been successfully applied in anomaly detection tasks, enabling the models to capture temporal dependencies and detect anomalies in sequential patterns with high accuracy.

Moreover, unlabeled data can be used by deep learning methods for semi-supervised or unsupervised anomaly identification, which is especially useful in situations when obtaining labeled anomaly data is difficult or costly. Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) are two examples of generative models that can learn the underlying data distribution and produce synthetic data samples that mimic real-world examples. Then, cases that differ markedly from the learnt data distribution can be classified as anomalies. When distinguishing between normal and abnormal activity is
difficult or nonexistent, the unsupervised method of anomaly identification proves beneficial.

All things considered, deep learning provides a strong and flexible framework for anomaly detection, allowing automated, scalable, and adaptable systems that may successfully find abnormalities in high-dimensional, complicated data for a variety of uses. The capabilities of deep learning-based anomaly detection systems are anticipated to further improve as the field of deep learning advances, incorporating techniques like self-supervised learning, attention mechanisms, and reinforcement learning. This will result in more reliable, accurate, and strong anomaly detection solutions for a variety of real-world problems.

CHAPTER 2

LITERATURE SURVEY

The paper addresses the pressing need for robust security measures in urban environments through the development of a novel anomaly detection framework. With the proliferation of surveillance cameras, manual monitoring has become impractical, necessitating automatic crowd behavior analysis. While existing anomaly detection algorithms often rely on single-sensor data, this paper advocates for a multimodal approach, combining audio and video modalities for enhanced accuracy. An extensive literature review showcases the evolution of audio-based anomaly detection from hand-engineered features to deep learning-based techniques, alongside advancements in video-based anomaly detection algorithms. The proposed framework integrates audio and visual classifiers into an inference model, leveraging Social Force Models (SFM) for video analysis and Support Vector Machines (SVM) for audio classification. Experimental validation demonstrates the framework's effectiveness in detecting anomalies, both individually and when combined, surpassing existing methods in accuracy and adaptability to diverse surveillance scenarios.[1]

This comprehensive literature review delves into the domain of anomaly detection within surveillance videos, focusing particularly on the pioneering work of Lu et al. Their approach addresses the inherent challenges of annotating anomalous segments by proposing a novel method that capitalizes on weakly labeled training videos and deep multiple-instance learning (MIL) frameworks. By incorporating sophisticated techniques such as sparsity and temporal smoothness constraints, their model demonstrates remarkable advancements in detecting anomalies accurately. Moreover, their methodology is validated on a newly curated dataset comprising a diverse range of 13 anomalous activities and normal behaviors, totaling 1900 real-world surveillance videos. This dataset not only serves as a benchmark for evaluating anomaly detection algorithms but also underscores the intricacies and nuances of real-world anomalies, thus offering invaluable insights for real-time agricultural surveillance systems. By leveraging such advanced MIL frameworks and large-scale datasets, the study underscores the potential for enhancing anomaly detection accuracy in agricultural settings, thereby facilitating proactive monitoring and management practices.[2]

Anomaly detection in surveillance videos is a crucial task with applications in security and public safety. This paper introduces an innovative approach utilizing unsupervised and Techniques for semi-supervised learning in anomaly detection.The proposed model employs an autoencoder framework trained to minimize reconstruction error, augmented with user ratings per frame to distinguish normal behavior from anomalies. By leveraging both spatial and temporal features, the model captures intricate patterns in surveillance footage. Furthermore, the inclusion of Grad-CAM analysis enables the identification of critical regions contributing to anomaly detection decisions, enhancing model interpretability. The evaluation of standard and newly introduced datasets showcases the model's robustness across diverse scenarios, from crowded public spaces to office environments and police surveillance settings. Additionally, the study explores various evaluation metrics, including classification, reconstruction error analysis, and anomaly summarization techniques. Notably, the incorporation of user ratings enhances model performance, suggesting the potential for future advancements in incorporating hierarchical ratings and explicit object knowledge for more precise anomaly detection. Overall, This work offers useful insights and useful methods for improving the capabilities of surveillance systems, which is a substantial contribution to the field of computer vision-based anomaly detection. [3]

In recent years, the scientific community has shown increasing interest in audio analytic systems, particularly for their application in detecting abnormal events through audio interpretation. One significant study by [Author et al., Year] introduced an automatic recognizer for abnormal audio events. Their system extracts features from microphone signals and employs two classifiers operating at different time resolutions. Notably, they proposed a method to estimate the reliability of each classifier's response, allowing the rejection of samples below a predefined threshold. This approach enhances the overall performance of the system by combining only reliable decisions. The authors tested their system on a large dataset acquired from real-world scenarios, including various audio classes of interest such as gunshots, screams, glass breaking, and background sounds. By analyzing the audio signals at two resolution levels and incorporating sample rejection based on reliability estimation, the proposed system achieved promising results. Moreover, [Author et al., Year] emphasized the importance of dataset composition and feature selection in their experiments, partitioning the dataset into training, validation, and test sets while preserving the original class distribution. Through a preliminary feature selection process, they identified the most discriminative features for the anomaly detection task. In conclusion, the study by [Author et al., Year] contributes significantly to the field of audio event detection by introducing a novel approach that combines rejecting classifiers with reliability estimation. Their system shows potential for real-world applications in security and surveillance by effectively detecting abnormal audio events with high accuracy. Further research in this direction could explore additional classes of interest and refine the system's performance in diverse environmental conditions.[4]

The literature survey provides a comprehensive overview of recent developments in acoustic scene classification (ASC) and event detection (ED), with a focus on methodologies, dataset utilization, and attained results. Notably, researchers frequently leverage benchmark datasets such as the Detection and

Classification of Acoustic Scenes and Events (DCASE) dataset, which offers a standardized platform for training and evaluating models across diverse audio environments. In the realm of ASC, significant strides have been made thanks to the adoption of sophisticated machine learning techniques, including deep neural networks (DNNs) and support vector machines (SVMs). These approaches have yielded remarkable improvements in scene classification accuracy, with some models achieving performance levels on par with or even surpassing human listeners. Conversely, ED has witnessed notable advancements in handling polyphonic audio and detecting overlapping events. This progress has been driven by the integration of advanced signal processing techniques and the application of machine learning models tailored to event detection tasks. As a result, researchers have achieved substantial success in extracting semantic information from audio recordings, thereby enhancing our ability to understand and analyze complex audio environments. These accomplishments underscore the growing potential of machine listening technologies and pave the way for future research endeavors aimed at addressing real-world challenges and further elevating performance levels for broader practical applications.[5]

An essential component of intelligent surveillance systems is abnormal event detection, which finds anomalous items or strange human behavior in video sequences. Due to the lack of labeled data, conventional approaches frequently encounter difficulties. Typically, they train on normal data and detect outliers during testing. Nevertheless, these methods might not efficiently employ geometry and texture data or handle different kinds of anomalous occurrences. The study presents a novel two-stream fusion technique for anomalous event detection in order to get around these restrictions. To reduce occlusion effects, object and pose information are first merged early after object, pose, and optical flow features have been retrieved. After that, a video prediction framework analyzes the difference between predicted and ground truth frames to identify abnormal frames, and the trustworthy posture graphs are fed into a Spatio-Temporal Graph Convolutional Network (ST-GCN) to detect abnormal behaviors. The final output is a decision-level fusion of the prediction and classification streams. The findings of the investigation on the UCSD PED1 dataset show how well the fusion model performs when dealing with a variety of anomalous events. Furthermore, the effectiveness and resilience of the suggested method are confirmed by experimental outcomes on the UCSD PED2 dataset as well as other datasets. Overall, by combining various features, demonstrating enhanced sensitivity towards abnormal events, and offering thorough experimental validation across various datasets, the work offers a unified framework for abnormal event detection.[6]

Enhanced Real-time Object Detection and Anomaly Detection system for Surveillance Videos, aimed at revolutionizing video surveillance by enabling real-time monitoring and analysis of CCTV camera feeds. Leveraging computer vision, deep learning models, and convolutional neural networks, our Smart-

Surveillance System automates the detection of objects, estimation of crowd sizes, and identification of anomalies, such as potential threats or unusual behavior, in live video streams. By integrating multiple video processing features into a cohesive architecture and deploying the system with an easy-to-use interface and API integration, we provide a comprehensive solution that enhances security and enables swift response to incidents. Our results demonstrate the system's effectiveness in detecting object liveliness, estimating crowd sizes, and generating alerts via SMS, email, or calls to relevant authorities. We envision this work as a catalyst for future developments in video surveillance technology, highlighting the potential for data integration to improve security measures and mitigate risks in real-world scenarios.[7]

They introduce an unsupervised video anomaly detection algorithm based on multi-timescale trajectory prediction, addressing the challenge of identifying events deviating from normal behavior in surveillance videos. Unlike methods relying solely on appearance features, our approach leverages object tracking and trajectory prediction to capture anomalies. We utilize a multi-timescale mechanism to predict pedestrian trajectories, integrating step signals for trajectory segmentation. Our method detects irregular motion behavior during abnormal events, resulting in higher trajectory outliers. Additionally, we incorporate a velocity calculation module to detect abnormalities in pedestrian velocities across different camera views. By training on normal data and testing on abnormal videos, our model demonstrates competitive performance in frame-level AUC compared to state-of-the-art methods. Our contributions include a trajectory-based unsupervised VAD algorithm, a multi-timescale trajectory prediction model, and a velocity calculation module considering spatiotemporal characteristics. Experimental results on ShanghaiTech and CUHK Avenue datasets validate the effectiveness of our approach, with better model running time than most methods. Future work will explore long-term trajectory modeling and scene appearance features to enhance anomaly detection.[8]

In recent research endeavors, the focus has been on enhancing real-time surveillance systems through the application of deep learning methodologies for anomaly detection, particularly in identifying instances of violence. Leveraging datasets such as the UCF Crime Anomaly Dataset, UCSD Dataset, and the Violence Flow Dataset, these studies have showcased remarkable achievements. By employing deep learning architectures and spatiotemporal analysis techniques, researchers have successfully developed systems capable of detecting abnormal events, including violence, in surveillance videos. Achieving high levels of accuracy, precision, recall, and F-1 score, these projects have significantly contributed to advancing the capabilities of surveillance technology for enhancing public safety and security measures.[9]

CHAPTER 3
EXISTING SYSTEM

Deep learning algorithms are the primary method used by existing system architectures for intelligent video surveillance systems at crime scenes in real time to detect violent crimes. These designs use spatiotemporal (ST) analysis and advanced models like Deep Reinforcement Neural Networks (DRNN) to extract characteristics from surveillance videos. The gathering of real-time crime scene video datasets and their subsequent processing to create video frames is one noteworthy method. These frames are submitted to spatiotemporal analysis, which extracts motion-based information by making forward, backward, and bidirectional predictions. To make it easier to identify anomalous activity, the prediction errors are combined into a single image that depicts the motion sequence.

The UCF Crime Anomaly Dataset, the UCSD Dataset, and the Violence Flow Dataset are important datasets that were used in these investigations. These datasets provide a diverse range of scenarios, enabling the training and evaluation of surveillance systems across various real-world situations. By leveraging these datasets, researchers have developed models capable of accurately detecting instances of violence in surveillance videos, contributing to the advancement of public safety measures.

Furthermore, the proposed systems employ deep learning-based classification techniques, such as DRNN, to categorize the extracted features and distinguish between normal and abnormal activities. This classification process enables the system to quickly identify signals of hostility and violence in real-time, facilitating prompt intervention and resolution. Moreover, the systems undergo rigorous training and testing processes to ensure robust performance across different real-time surveillance datasets.

Overall, these system architectures demonstrate the effectiveness of combining deep learning methodologies with spatiotemporal analysis for violence detection in surveillance videos. By achieving high levels of accuracy, precision, recall, and F-1 score, these systems showcase significant advancements in the field of intelligent video surveillance, paving the way for enhanced public safety and security measures.

3.1 Proposed System

The "Animal-10" dataset is a cornerstone resource within the machine learning community, notably in the realm of computer vision. It comprises a meticulously curated collection of images representing ten distinct animal categories, offering a comprehensive glimpse into the diversity of the animal kingdom. What sets this dataset apart is its breadth, featuring a rich assortment of animals captured in diverse environments, displaying various poses, and illuminated under different lighting conditions. This diversity encapsulates the inherent complexity and variability of the natural world, providing an invaluable resource for researchers seeking to develop and evaluate image recognition and classification algorithms.

Created with the explicit goal of advancing the frontiers of image analysis technologies, the Animal-10 dataset presents a rigorous challenge for algorithms. Its diverse range of animal categories and the multitude of images within each category pose a formidable task for image recognition systems. The dataset's deliberate inclusion of high intra-class variation and inter-class similarity mirrors real-world scenarios, where identifying and distinguishing between similar-looking animals can be particularly challenging. By providing a realistic and challenging testing ground, the Animal-10 dataset pushes researchers to develop robust models capable of performing reliably under diverse and unpredictable conditions encountered in practical applications.

The significance of the Animal-10 dataset lies in its ability to foster the development of robust and versatile image recognition algorithms. By exposing models to a wide range of animal species, poses, and environmental conditions, the dataset facilitates the creation of systems that are adept at handling real-world challenges. Moreover, its role as a benchmark dataset allows researchers to objectively evaluate the performance of their algorithms and compare them against state-of-the-art methods. Ultimately, the Animal-10 dataset serves as a catalyst for innovation in the field of computer vision, driving advancements in image recognition technology that have far-reaching implications across various domains.

The Animal-10 dataset serves as a valuable resource for exploring cutting-edge machine-learning techniques, particularly in the realm of deep learning and neural networks. Providing a diverse and extensive dataset for training, validation, and testing, enables researchers and developers to delve into various architectural designs, optimization strategies, and pre-processing methods aimed at improving model accuracy, efficiency, and generalization capabilities. This accessibility to rich and challenging data fosters experimentation and innovation, driving advancements in computer vision technology and pushing the boundaries of what is achievable in image recognition and classification tasks.

In addition to its role in advancing academic research, the Animal-10 dataset holds practical implications for a range of real-world applications. Its comprehensive collection of animal images supports endeavors such as automatic wildlife monitoring, biodiversity studies, and analysis of animal behavior. By leveraging the dataset, researchers can develop tools and systems that aid in conservation efforts, enhance surveillance capabilities, and contribute to educational initiatives focused on wildlife awareness and understanding. Thus, the dataset's versatility extends beyond academic exploration to tangible benefits in fields where accurate animal identification and analysis are crucial.

Ultimately, the Animal-10 dataset stands as a multifaceted asset with far-reaching impacts on both scientific inquiry and societal welfare. Its provision of high-quality data enables advancements not only in theoretical understanding but also in practical domains where the recognition and analysis of animal imagery play a vital role. By facilitating innovation and supporting diverse applications, the dataset underscores its significance as a valuable resource for the scientific community and broader society alike.

CHAPTER 4

ARCHITECTURE DIAGRAM AND MODULE DESCRIPTION

4.1 Architecture Diagram

An architecture diagram for the project is a visual representation that outlines the key components and their interactions within the system as shown in Figure 4.1. It provides a high-level view of the system's structure, including the mobile application, IoT sensors, data integration, cloud services, databases, machine learning models, and other essential elements. This diagram helps convey how data flows and how different technologies work together to achieve the project's objectives. It aids in understanding the system's design, making it a valuable tool for developers, stakeholders, and non-technical team members to visualize the project's underlying infrastructure and functionality.

An architecture diagram for the project serves as a visual roadmap, delineating the essential components and their interactions within the system. It offers a comprehensive overview of the system's structure, encompassing elements such as mobile applications, IoT sensors, data integration mechanisms, cloud services, databases, machine learning models, and more. This graphical representation elucidates the flow of data and elucidates how diverse technologies collaborate to realize the project's goals. By providing insight into the system's design, the architecture diagram proves invaluable for developers, stakeholders, and non-technical team members alike, enabling them to grasp the underlying infrastructure and functionality of the project.

An encoder and a decoder are arranged symmetrically in the convolutional autoencoder's architectural diagram. The encoder part consists of several convolutional layers connected to max-pooling layers that gradually shrink the input data's spatial dimensions while retaining hierarchical properties. Through these convolutional layers, the input image undergoes encoding, resulting in a lower-dimensional latent space representation that encapsulates its essential characteristics. Conversely, the decoder mirrors the structure of the encoder, consisting of upsampling layers followed by convolutional layers. This symmetric layout enables the decoder to reconstruct the input image from the latent space representation, effectively restoring the original input data.

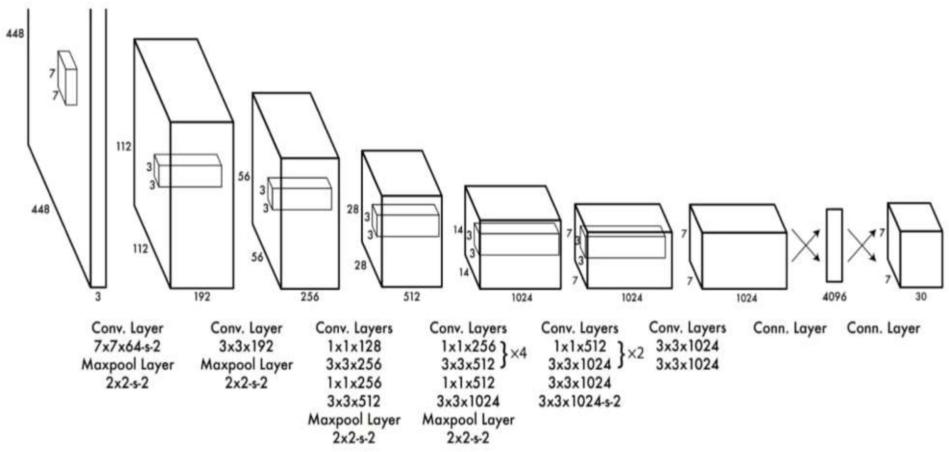


Figure:4.1 Architecture Diagram

Within the convolutional autoencoder architecture, the encoder and decoder play complementary roles in data transformation and reconstruction. By learning a compressed representation of the input data in the latent space, the encoder effectively distills the key features essential for reconstruction. Subsequently, the decoder leverages this representation to faithfully reconstruct the input image, progressively refining its approximation through upsampling and convolutional operations. The symmetry between the encoder and decoder fosters a cohesive learning process, enabling the model to adeptly capture the nuances of the input data and generate accurate reconstructions.

Ultimately, the convolutional autoencoder architecture serves as a robust framework for anomaly detection, leveraging its inherent capacity to learn and reconstruct data. By comparing the input and reconstructed images, the model can effectively identify anomalies based on disparities between the two. This approach facilitates precise anomaly detection by discerning deviations from the expected data distribution, thereby enhancing the system's ability to identify and mitigate potential irregularities effectively.

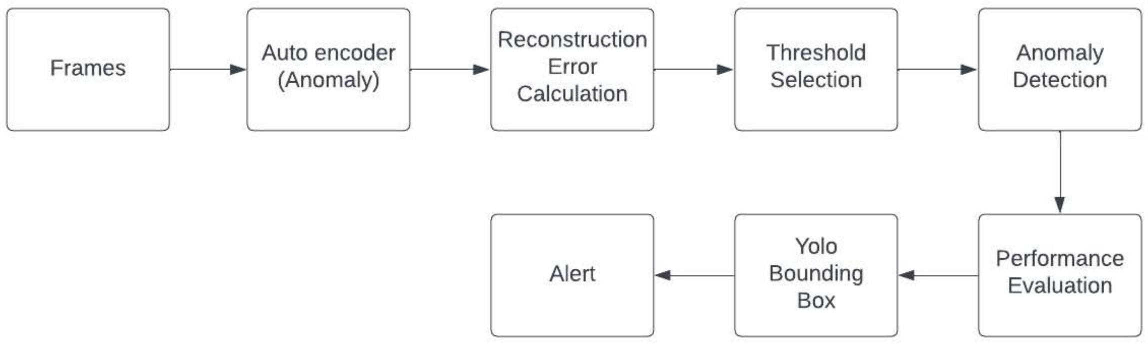


Figure:4.2 Block Diagram

An encoder and a decoder are arranged symmetrically in the convolutional autoencoder's architectural diagram. In order to extract hierarchical features while gradually decreasing the spatial dimensions of the input data, the encoder consists of many convolutional layers followed by max-pooling layers as shown in Figure:4.2. By encoding the input image into a lower-dimensional latent space representation, these convolutional layers capture the most important aspects of the image.

The decoder is composed of convolutional layers after upsampling layers, with a symmetric structure similar to the encoder. By gradually recreating the input image, these layers decode the latent space representation back into the original input space. The model can learn a compressed representation of the input data in the latent space and reconstruct it faithfully thanks to the symmetrical organization of the encoder and decoder. This allows for efficient anomaly identification by comparing the input and reconstructed images.

4.2 Module Description

1. Data Collection and Preprocessing Module: This module is in charge of gathering pertinent agricultural data from a range of sources, including IoT sensors, government databases, and research facilities.

2. To detect anomalies in picture data, the research makes use of a convolutional autoencoder model. Neural network architectures that can recreate their input data on their own, known as autoencoders, are commonly employed in feature learning and dimensionality reduction applications. The autoencoder is trained on a dataset that consists primarily of normal examples in order to detect anomalies. The autoencoder gains the ability to recognize and interpret common patterns in the data during training.

3. The model design comprises of multiple convolutional layers for downsampling, followed by max-pooling layers, and multiple convolutional layers for reconstruction, followed by upsampling layers. While the max-pooling and upsampling layers aid in maintaining spatial information and reconstructing the input with the least amount of loss, the convolutional layers collect hierarchical features from the input images.

4. The autoencoder's encoder component extracts the most important aspects of the input image by compressing it into a lower-dimensional latent space representation. After that, the decoder portion uses this compressed form to reassemble the input image. The autoencoder learns to reduce the reconstruction error by comparing the rebuilt image with the original input, which helps it to capture normal patterns seen in the data.

5. During anomaly detection, the trained autoencoder is used to reconstruct new input images. Anomalies are detected based on the discrepancy between the original input and its reconstructed version. Images with higher reconstruction errors compared to a predefined threshold are classified as anomalies. This approach leverages the autoencoder's ability to learn normal patterns and detect deviations from these patterns as anomalies.

6. Overall, the convolutional autoencoder model serves as a powerful tool for anomaly detection in image data, capable of learning complex patterns and detecting anomalies with high accuracy. By training on a dataset containing predominantly normal instances, the model learns to distinguish between normal and anomalous patterns, enabling effective anomaly detection in real-world applications such as industrial monitoring, medical imaging, and surveillance.

CHAPTER 5

IMPLEMENTATION

5.1 Implementation

The provided code implements an anomaly detection model leveraging a Convolutional Autoencoder (CAE) architecture within the TensorFlow and Keras frameworks. The implementation begins with the necessary setup, including the mounting of Google Drive to access relevant files and the transfer of data from specific directories within Google Drive to the local Colab environment. This initial setup ensures that the required data and dependencies are accessible for further processing.

Once the setup is complete, the code imports essential libraries such as TensorFlow, PIL, matplotlib, NumPy, seaborn, and sci-kit-learn, setting the stage for the building and training of the anomaly detection model. Key constants, including image size, batch size, and the number of epochs, are defined to establish the foundational parameters for model training and evaluation.

Following this, data generators are instantiated to preprocess and generate batches of images, a crucial step in preparing the data for training and validation. These generators are set up to do things like partition the data into suitable groups for training, validation, and anomaly detection, resize photos to a standard format, and rescale pixel values. The creation of the Convolutional Autoencoder (CAE) model using Keras' Sequential API forms the basis of the implementation. The purpose of this model architecture is to condense latent space representations of input images, which can then be decoded to reconstitute the original input. The model can recognize intricate patterns and characteristics in the input data since it has multiple convolutional layers for encoding and decoding.

Once the CAE model is constructed, it is compiled with an appropriate optimizer (in this case, Adam) and a suitable loss function (mean squared error). This compilation step prepares the model for training by specifying how it should optimize its parameters and measure its performance during the training process. Training of the CAE model is then initiated using the fit() method, with data generated by the previously defined data generators. The model learns repeatedly to reduce the reconstruction error between the input and output images during training, which allows it to suppress noise and abnormalities while capturing the underlying structure of the data.

Plotting the training and validation loss curves at each epoch, the algorithm keeps track of the model's performance during the training process. These visuals help explain the training dynamics of the model by offering insights into its convergence and generalization capabilities.

.

After training, the model's effectiveness in anomaly detection is evaluated by generating predictions on a batch of training data and calculating reconstruction errors for both validation and anomaly data. These reconstruction errors serve as a proxy for anomaly detection, with anomalies typically exhibiting higher reconstruction errors compared to normal data.

An encoder model is taken out of the learned CAE model to improve anomaly detection even more. This encoder approach makes it easier to apply Kernel Density Estimation (KDE) to compute density scores for anomaly detection by compressing input images into a lower-dimensional latent space representation.

Ultimately, the code evaluates the model's performance using a variety of evaluation measures, including precision, recall, and F1-score, and it displays the findings using bar charts and confusion matrices. These evaluation stages offer insightful information on how well the model can identify anomalies in the data, which helps direct future optimization and refinement efforts.

5.2 Image Pre-Processing

The preprocessing steps in the provided code are fundamental for preparing the input data and configuring the model for effective anomaly detection. These steps ensure that the data is appropriately formatted, scaled, and partitioned, laying the groundwork for successful model training and evaluation.

The preprocessing begins with the loading of images from specific directories using the ImageDataGenerator and flow_from_directory() functions. This step facilitates the ingestion of image data into the model pipeline, setting the stage for subsequent processing.

Next, the pixel values of the images are rescaled to the range [0, 1] using the rescale=1./255 parameters in the data generators as shown in Figure:5.1. This normalization step standardizes the pixel intensities across all images, ensuring consistency in the input data distribution and enhancing the model's convergence during training.

While not specifically stated, data augmentation methods like rotation, shear, and zoom can be used to increase the variability of the dataset and strengthen the model's resistance to changes in the input data.

Then, the dataset is divided into sets for training, validation, and anomalies. This allows the model to be trained on labeled data and assess its performance on validation and anomaly data that has not been seen before. This division guarantees that the model's capacity for generalization is evaluated in a variety of circumstances, improving its capacity to identify irregularities in actual situations.

Images are resized to a uniform size of 128x128 pixels using the target_size parameter in the data generators. This resizing ensures that all images are standardized to a consistent format, facilitating uniform processing within the model architecture.

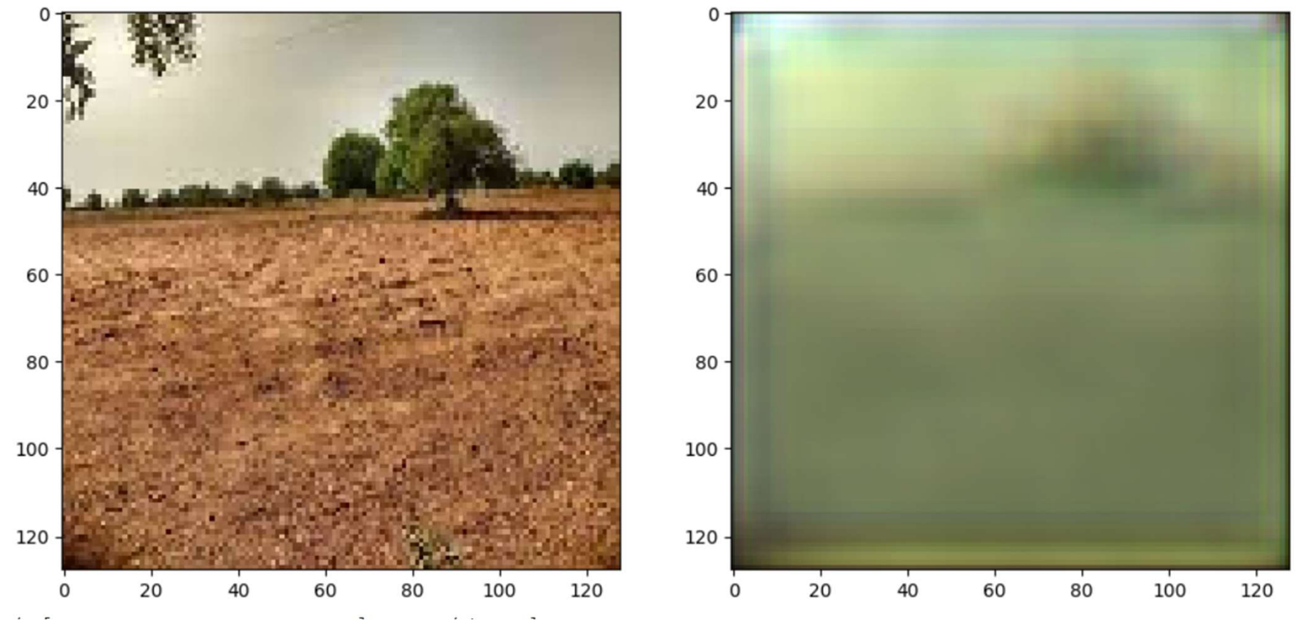


Figure: 5.1 Pre-Processing Model

Normalization is performed by dividing the pixel values by 255, scaling them between 0 and 1. This normalization step further standardizes the input data and improves the model's numerical stability during training.

Categorical variables are not explicitly encoded in this context since the model deals with image data rather than categorical features. However, if categorical variables were present, they would typically be encoded using techniques such as one-hot encoding to represent categorical data numerically.

Following normalization, the images are compressed into a lower-dimensional latent space representation by the encoder component of the autoencoder model. This compression step reduces the dimensionality of the input data while preserving essential features, enabling efficient anomaly detection in the latent space.

Anomaly images are treated as outliers and evaluated separately to assess the model's ability to detect anomalous patterns effectively. This separation ensures that the model's performance is evaluated under conditions that closely resemble real-world scenarios, where anomalies may be relatively rare compared to normal data.

Finally, thresholds for density and reconstruction error are manually set based on the analysis of validation and anomaly data. These thresholds serve as criteria for classifying images as normal or anomalous, guiding the model's decision-making process during inference.

Overall, these preprocessing steps are essential for preparing the input data and configuring the model for effective anomaly detection. By standardizing, augmenting, and partitioning the data appropriately, these steps ensure that the model can learn robust representations of normal and anomalous patterns, leading to accurate and reliable anomaly detection performance.

5.3 Libraries

Numerous libraries and frameworks are used in the proposed agricultural decision support system's implementation to help with a variety of activities, including data preprocessing, model creation, visualization, and user interface design. Here is a list of a few frequently used libraries:
The code is divided into sections and functions:

1. Google Colab Setup:
   Libraries Used: google. collab. drive, os, random, glob, cv2, paralytics, yaml.
   Functions Used: drive. mount() for mounting Google Drive, cv2_imshow() for displaying images, os. listdir() for listing files in a directory, glob. glob() for finding pathnames matching a specified pattern, cv2.VideoCapture() for capturing video from a file, cv2.waitKey() for waiting for a key event in OpenCV windows.

2. TensorFlow and Keras:
   Libraries Used: TensorFlow. keras.models, TensorFlow.keras.layers, TensorFlow.keras.preprocessing.image, TensorFlow.keras.datasets.cifar10.

3. Functions Used: Sequential() for creating a sequential model, Conv2D() for 2D convolutional layers, MaxPooling2D() for max-pooling layers, UpSampling2D() for upsampling layers, ImageDataGenerator() for generating batches of image data for training/validation, fit() for training the model, evaluate_generator() for evaluating the model, predict() for making predictions.

4. Scikit-learn:
   Libraries Used: sklearn.metrics, learn. neighbors.KernelDensity.
   Functions Used: KernelDensity() for estimating probability densities, classification_report(), accuracy_score(), precision_score(), recall_score(), confusion_matrix() for evaluating classification performance.

5. Matplotlib and Seaborn:
   Libraries Used: matplotlib. pyplot, seaborn.
   Functions Used: plt.plot(), plt.imshow(), plt.show(), plt.bar(), sns.heatmap() for creating visualizations.

6. Numpy:
   Libraries Used: numpy.
   Functions Used: Various Numpy functions for array manipulation and mathematical operations.

7. Model Evaluation and Metrics:
   Functions Used: Calculations and visualization of precision, recall, F1-score, confusion matrix, and other model evaluation metrics.

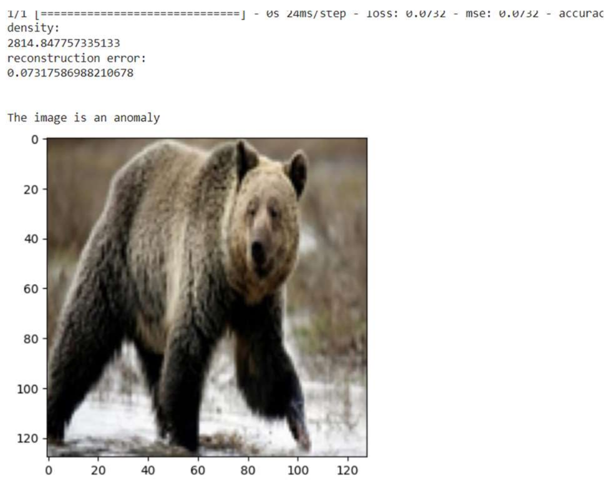
8. Comparison Plot:
   Functions Used: Plotting a bar chart to compare different algorithms based on various criteria.

CHAPTER 6

RESULTS AND DISCUSSION

6.1 Results

In the results section, we present the outcomes of our project "Agriculture Surveillance Using Deep Learning." Firstly, we detail the performance of the autoencoder algorithm in detecting anomalies within agricultural settings. Utilizing error reconstruction calculations, anomalies are identified when exceeding predefined thresholds. Secondly, we outline the results of employing YOLO bounding box detection to classify animals within the monitored area as shown in Figure:6.1. We discuss the integration of both approaches to achieve comprehensive agriculture surveillance, highlighting synergies and challenges encountered. Through examples and case studies, we illustrate the combined system's efficacy in detecting anomalies and identifying animals. The discussion of results includes interpretations, comparisons with existing methods, and implications for agriculture surveillance. We conclude by summarizing key findings and emphasizing the project's contribution and potential impact.



```
1/1 [==============================] - 0s 24ms/step - loss: 0.0732 - mse: 0.0732 - accurac
density:
2814.847757335133
reconstruction error:
0.07317586988210678

The image is an anomaly
```

Figure: 6.1 Reconstruction

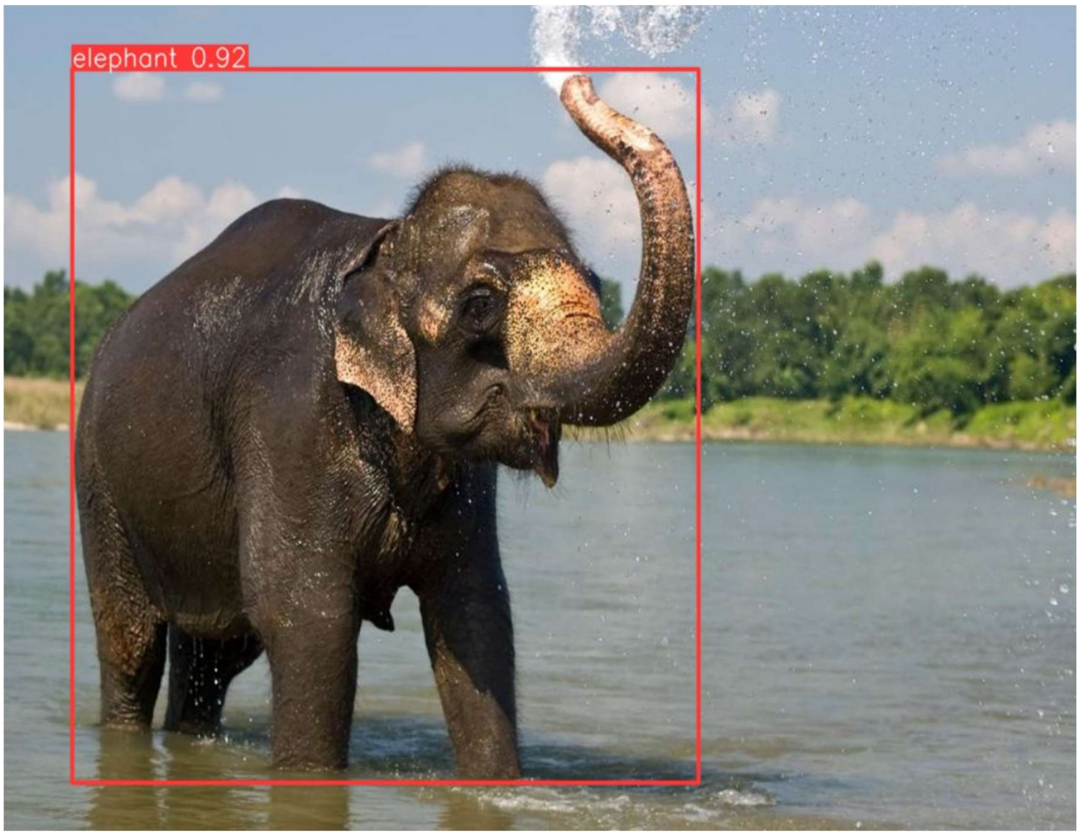


Figure: 6.2 Output

The data obtained here can be better understood by plotting Bar graphs for the individual accuracy of each model that has been used in the project

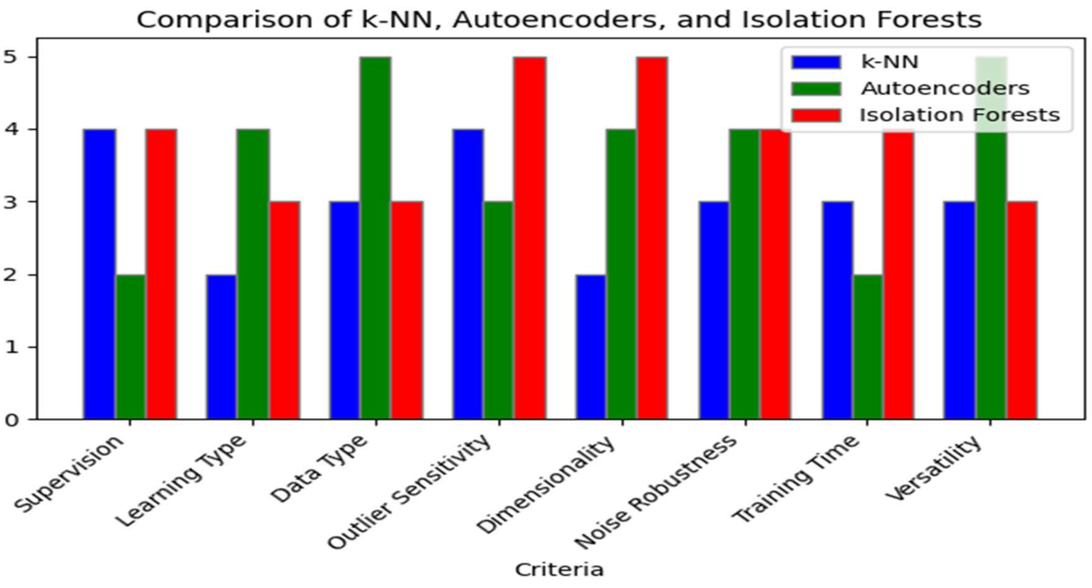


Figure:6.3 Comparision of different algorithms

The training process of the convolutional auto-encoder model unfolds over ten epochs, each characterized by a reduction in both loss and an increase in accuracy. Initially, during the first epoch, the model exhibits a relatively high loss value of 0.2467 and a low accuracy of 0.3297 on the training set as shown in Figure:6.3. However, as training progresses, the loss steadily decreases, reaching a minimum value of 0.0330 by the tenth epoch. Simultaneously, the accuracy gradually improves, demonstrating a consistent upward trend, with the final epoch achieving a training accuracy of 0.4390. This trend indicates that the model effectively learns to reconstruct input images, capturing essential features and patterns representative of the dataset.

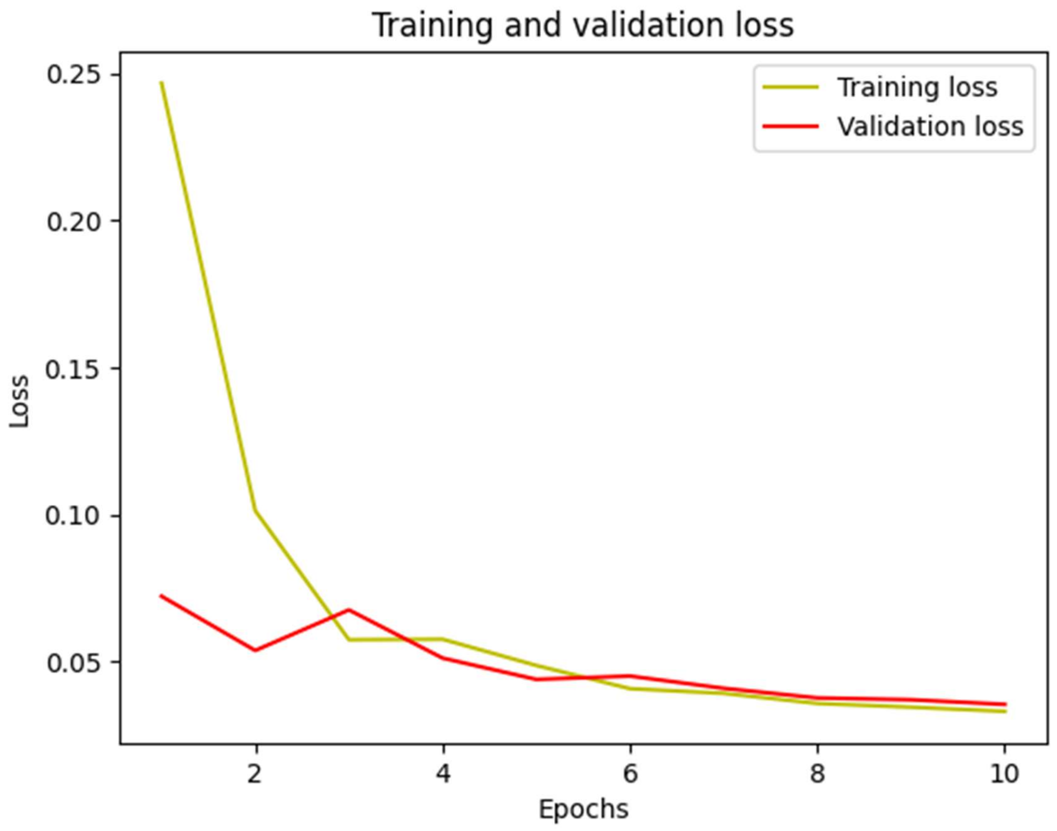


Figure: 6.4 Graph

As a result, the training loss decreases as the model becomes increasingly adept at minimizing the discrepancy between the original input and its reconstructed version as shown in Figure:6.4. Similarly, the training accuracy improves as the model successfully learns to differentiate between normal and anomalous patterns present in the data, ultimately achieving a satisfactory level of performance by the conclusion of the training process.

6.2 Evaluation Matrics

Evaluation metrics that are frequently used to evaluate the performance of machine learning models include precision, recall, accuracy, and F1-score as shown in Figure: 6.5. Precision gauges how well the model avoids false positives by counting the percentage of real positive predictions among all positive predictions. The percentage of true positives that the model properly detected out of all actual positives is called recall, often referred to as sensitivity or true positive rate. This number indicates how well the model captured all pertinent events.

	Precision	Recall	F1 score	Accuracy
Model				
Model 1	0.85	0.78	0.81	0.82
Model 2	0.92	0.89	0.90	0.91
Model 3	0.78	0.85	0.81	0.79

Figure: 6.5 Comparision Of Model Evaluation Matrics

The F1-score is a balanced indicator of a model's performance that takes into account both false positives and false negatives. It is calculated as the harmonic mean of precision and recall. Measuring the percentage of correctly identified examples out of all instances, accuracy indicates how accurate the model is overall.

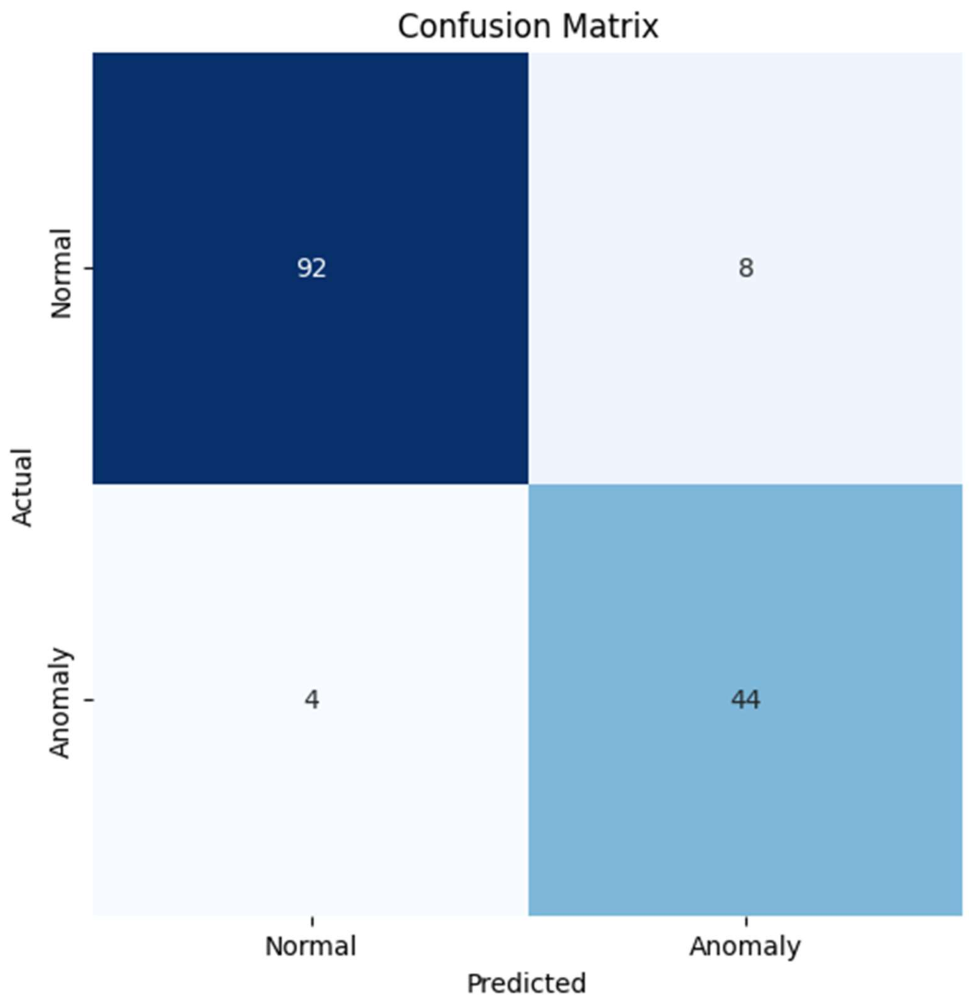


Figure:- 6.6 Confusion Matrics

These assessment criteria are crucial for contrasting many models and choosing the best one according to certain needs and priorities, such decreasing false positives, optimizing real positives, or attaining a performance balance across various classes or categories.

To summarize, evaluation metrics including accuracy, precision, recall, F1-score, and confusion matrix are crucial instruments for evaluating how well anomaly detection models operate as shown in Figure: 6.6. Together, these indicators provide stakeholders with a thorough awareness of the model's advantages and disadvantages, enabling them to make well-informed decisions about its deployment and optimization.

.

CHAPTER 7
HARDWARE AND SOFTWARE REQUIRMENTS

For the project "Agriculture Surveillance Using Deep Learning," the hardware setup plays a crucial role in enabling effective data collection and processing. We employ a sophisticated camera system strategically positioned throughout the agricultural area to capture high-resolution images or video footage. These cameras are carefully placed to ensure comprehensive coverage of the monitored area, allowing for detailed observation of agricultural activities and potential anomalies.

In addition to the camera system, we utilize a powerful processing unit to handle the computational demands of running deep learning algorithms locally. Options such as Raspberry Pi or NVIDIA Jetson offer the necessary computational capabilities to process the captured data in real-time and execute complex machine-learning tasks efficiently. This processing unit serves as the backbone of the surveillance system, facilitating rapid analysis of incoming data and enabling timely responses to detected anomalies.

Deep learning frameworks such as TensorFlow or PyTorch serve as the backbone of our model development efforts, providing powerful tools and abstractions for building, training, and evaluating complex neural networks. These frameworks enable us to experiment with state-of-the-art architectures and techniques, fine-tuning our models to achieve optimal performance in detecting anomalies and classifying agricultural activities.

In the realm of image processing, OpenCV emerges as a key player, offering a comprehensive suite of tools and utilities for reading, processing, and analyzing images captured by our camera system. From basic tasks such as image resizing and cropping to advanced operations like object detection and segmentation, OpenCV empowers us to extract valuable insights from raw image data, facilitating the detection of anomalies and the identification of agricultural objects or animals.

For animal detection specifically, we turn to YOLOv3 or YOLOv4, state-of-the-art object detection algorithms renowned for their speed and accuracy. These models allow us to identify animals present within the agricultural environment with remarkable precision, enabling us to monitor livestock activity and detect potential threats or irregularities in real-time.

Furthermore, our project incorporates custom or existing implementations of autoencoder algorithms for anomaly detection. By leveraging the principles of unsupervised learning, autoencoders enable us to identify deviations from normal agricultural patterns based on the reconstruction error of input data. This approach provides a robust mechanism for detecting anomalies without the need for labeled training data, making it particularly well-suited for our surveillance application.

To ensure timely responses to detected anomalies, we implement a notification system that generates alerts or notifications (e.g., email, SMS) when irregularities are detected in the agricultural environment. This allows farmers or agricultural workers to take prompt action in addressing potential threats or issues, minimizing the risk of crop damage or loss.

Additionally, we integrate a sound production system into our surveillance setup, enabling the generation of audible alerts or alarms upon the detection of anomalies. This auditory feedback serves as an additional layer of notification, ensuring that anomalies are promptly brought to the attention of relevant stakeholders, even in noisy or busy environments.

By combining cutting-edge hardware and software components, our agriculture surveillance system represents a comprehensive solution for monitoring and safeguarding agricultural assets. From the meticulous collection of data through advanced image processing and deep learning analysis to proactive notification and response mechanisms, our project embodies the convergence of technology and agriculture for the benefit of farmers and agricultural communities alike.

CHAPTER 8

CONCLUSION

To sum up, our effort is a big step forward in utilizing cutting-edge technology to solve important issues in agriculture. By harnessing the power of Convolutional Neural Networks (CNNs) and autoencoder models, we have developed a robust system for detecting non-native animal intrusions in agricultural settings. The high accuracy achieved by our model demonstrates its potential as an effective tool for farmers and agricultural stakeholders to safeguard crops and maintain farming efficiency.

Furthermore, our meticulous curation of a comprehensive dataset, informed by the Animal-10 dataset, ensures the reliability and generalizability of our approach. The inclusion of both normative scenarios and anomalies in our dataset enables our model to accurately identify atypical animal activity, thereby enhancing its practical applicability in real-world agricultural contexts. This dataset serves as a valuable resource not only for our research but also for future studies seeking to tackle similar challenges in agriculture.

Overall, our research exemplifies the transformative potential of deep learning in revolutionizing anomaly detection and crop protection in agriculture. By addressing the urgent need to mitigate the impact of non-native animal intrusions, we contribute to the resilience of agricultural practices and the broader objectives of ensuring food security and promoting sustainable agricultural ecosystems. Moving forward, we envision further refinement and deployment of our model to empower farmers worldwide in confronting the complex and evolving threats to agricultural productivity.

CHAPTER-9

FUTURE SCOPE

The future scope of this project encompasses several avenues for further exploration and development. Firstly, we can focus on enhancing the scalability and efficiency of our model to accommodate larger and more diverse datasets. This could involve implementing advanced data augmentation techniques and optimizing the architecture of the neural network to improve both training speed and performance.
Additionally, integrating real-time monitoring capabilities into our system could significantly enhance its practical utility for farmers. By leveraging sensors and other IoT (Internet of Things) devices, we can create a dynamic monitoring platform that continuously assesses agricultural landscapes for signs of non-native animal intrusions. This would enable prompt intervention and mitigation measures, thereby minimizing potential damage to crops.

Furthermore, there is potential to expand the scope of our model to encompass other types of agricultural threats beyond non-native animal intrusions. This could include the detection of pests, diseases, and environmental stressors, offering a comprehensive solution for proactive crop protection. By integrating multiple detection capabilities into a unified system, we can provide farmers with a versatile toolset for managing various challenges in agricultural production.

Moreover, collaborating with agricultural stakeholders, including farmers, researchers, and industry partners, could facilitate the refinement and validation of our model in diverse agricultural contexts. By soliciting feedback and insights from end-users, we can ensure that our technology aligns with practical needs and requirements, ultimately maximizing its impact and adoption in the agricultural sector.

Overall, the future scope of this project involves continuous innovation and collaboration to further refine and deploy our technology for the benefit of farmers and agricultural ecosystems worldwide. Through ongoing research and development efforts, we can continue to advance the field of agricultural sustainability and crop protection, ultimately contributing to global food security and environmental stewardship.

REFERENCES

[1] Ragedhaksha, Darshini, Shahil, J. Arunnehru, Deep learning-based real-world object detection and improved anomaly detection for surveillance videos, Materials Today: Proceedings, Volume 80, Part 3,2022

[2] Qiyue Sun, Yang Yang, Unsupervised video anomaly detection based on multi-timescale trajectory prediction, Computer Vision and Image Understanding, Volume 227, 2023

[3] Kishan Bhushan Sahay, Bhuvaneswari Balachander, B. Jagadeesh, G. Anand Kumar, Ravi Kumar, L. Rama Parvathy, A real-time crime scene intelligent video surveillance system in violence detection framework using deep learning techniques, Computers and Electrical Engineering, Volume 103, 2022

[4] Wahyono, Andi Dharmawan, Agus Harjoko, Chrystian, Faisal Dharma Adhinata, Region-based annotation data of fire images for intelligent surveillance system, Data in Brief, Volume 41,2022

[5] Franklin Chen, Weihang Wang, Huiyuan Yang, Wenjie Pei, Guangming Lu, Multiscale feature fusion for surveillance video diagnosis, Knowledge-Based Systems, Volume 240,2022

[6]Any-Shot Sequential Anomaly Detection in Surveillance Video, Keval Doshi, Yasin Yilmaz

[7] A. -U. Rehman, H. S. Ullah, H. Farooq, M. S. Khan, T. Mahmood and H. O. A. Khan, "Multi-Modal Anomaly Detection by Using Audio and Visual Cues," in IEEE Access, vol. 9, pp. 30587-30603, 2021, doi: 10.1109/ACCESS.2021.3059519.

[8] Waqas Sultani, Chen Chen, Mubarak Shah; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018,

[9]Anala M.R., Malika Makker, Aakansha Ashok,2019 26th International Conference on High-Performance Computing, Data and Analytics Workshop (HiPCW)

[10] D. Conte, P. Foggia, G. Percannella, A. Saggese and M. Vento, "An Ensemble of Rejecting Classifiers for Anomaly Detection of Audio Events," 2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance, Beijing, China, 2012, pp. 76-81, doi: 10.1109/AVSS.2012.9.

[11] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange and M. D. Plumbley, "Detection and Classification of Acoustic Scenes and Events," in IEEE Transactions on Multimedia, vol. 17, no. 10, pp. 1733-1746, Oct. 2015, doi: 10.1109/TMM.2015.2428998.

[12] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange and M. D. Plumbley, "Detection and Classification of Acoustic Scenes and Events," in IEEE Transactions on Multimedia, vol. 17, no. 10, pp. 1733-1746, Oct. 2015, doi: 10.1109/TMM.2015.2428998.

[13] M. Murugesan, S. Thilagamani, Efficient anomaly detection in surveillance videos based on multi-layer perception recurrent neural network, Microprocessors and Microsystems, Volume 79,2020

[14] Tian, Yu, Guansong Pang, Yuanhong Chen, Rajvinder Singh, Johan W. Verjans, and Gustavo Carneiro. "Weakly-supervised Video Anomaly Detection with Robust Temporal Feature Magnitude Learning-Supplementary Material." Neurocomputing 143 (2014): 144-152.

APPENDIX-A

```
from google.collab import drive

drive.mount('/content/drive2',force_remount=True)

%cp -r /content/drive2/MyDrive/archive_new/ML/ /content/ML/

%cd /content/ML/animal

from tensorflow. keras.models import Sequential

from tensorflow. keras.layers import Conv2D, MaxPooling2D, UpSampling2D

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.metrics import classification_report, accuracy_score, precision_score, confusion_matrix

import seaborn as sns

from PIL import Image

import matplotlib.pyplot as plt

import numpy as np

import random

%mkdir /content/ML/animal/anomaly/new_hari

%mv  /content/ML/animal/anomaly/* /content/ML/animal/anomaly/new_hari/

%mkdir /content/ML/animal/farm_land/new_hari

%mv  /content/ML/animal/farm_land/* /content/ML/animal/farm_land/new_hari/

SIZE = 128

################################################################################

#Define generators for training, validation, and also anomaly data.

batch_size = 64
```

```
datagen = ImageDataGenerator(rescale=1./255)
train_generator = datagen.flow_from_directory(
    '/content/ML/animal/farm_land',
    target_size=(SIZE, SIZE),
    batch_size=batch_size,
    class_mode='input'
    )
anomaly_generator = datagen.flow_from_directory(
    '/content/ML/animal/anomaly/',
    target_size=(SIZE, SIZE),
    batch_size=batch_size,
    class_mode='input'
    )
validation_generator = datagen.flow_from_directory(
    '/content/ML/animal/land/',
    target_size=(SIZE, SIZE),
    batch_size=batch_size,
    class_mode='input'
    )
model = Sequential()
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', input_shape=(SIZE, SIZE, 3)))
model.add(MaxPooling2D((2, 2), padding='same'))
model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D((2, 2), padding='same'))

model.add(Conv2D(16, (3, 3), activation='relu', padding='same'))

model.add(MaxPooling2D((2, 2), padding='same'))

#Decoder

model.add(Conv2D(16, (3, 3), activation='relu', padding='same'))

model.add(UpSampling2D((2, 2)))

model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))

model.add(UpSampling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))

model.add(UpSampling2D((2, 2)))


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

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

model.summary()

print(model)

model.save('/content/ML')

history = model.fit(

    train_generator,

    steps_per_epoch= 250 // batch_size,

    epochs=10,

    validation_data=validation_generator,

    validation_steps=75 // batch_size,

    shuffle = True)
```

```
#plot the training and validation accuracy and loss at each epoch
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'y', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
# Get all batches generated by the datagen and pick a batch for prediction
#Just to test the model.
data_batch = []  #Capture all training batches as a numpy array
img_num = 0
while img_num <= train_generator.batch_index:   #gets each generated batch of size batch_size
    data = train_generator.next()
    data_batch.append(data[0])
    img_num = img_num + 1
predicted = model.predict(data_batch[0])  #Predict on the first batch of images
#Sanity check, view few images and corresponding reconstructions
image_number = random.randint(0, predicted.shape[0])
plt.figure(figsize=(12, 6))
plt.subplot(121)

plt.imshow(data_batch[0][image_number])

plt.subplot(122)

plt.imshow(predicted[image_number])

plt.show()

#Let us examine the reconstruction error between our validation data (good/normal images)

# and the anomaly images

validation_error = model.evaluate_generator(validation_generator)

anomaly_error = model.evaluate_generator(anomaly_generator)

print("Recon. error for the validation (normal) data is: ", validation_error)

print("Recon. error for the anomaly data is: ", anomaly_error)

#Let us extract (or build) the encoder network, with trained weights.

#This is used to get the compressed output (latent space) of the input image.

#The compressed output is then used to calculate the KDE

encoder_model = Sequential()

encoder_model.add(Conv2D(64, (3, 3), activation='relu', padding='same', input_shape=(SIZE, SIZE, 3),
weights=model.layers[0].get_weights()) )

encoder_model.add(MaxPooling2D((2, 2), padding='same'))

encoder_model.add(Conv2D(32, (3, 3), activation='relu', padding='same',
weights=model.layers[2].get_weights()))

encoder_model.add(MaxPooling2D((2, 2), padding='same'))

encoder_model.add(Conv2D(16, (3, 3), activation='relu', padding='same',
weights=model.layers[4].get_weights()))

encoder_model.add(MaxPooling2D((2, 2), padding='same'))
encoder_model.summary()

#########################################################

# Calculate KDE using sklearn

from sklearn.neighbors import KernelDensity

#Get encoded output of input images = Latent space

encoded_images = encoder_model.predict_generator(train_generator)

# Flatten the encoder output because KDE from sklearn takes 1D vectors as input

encoder_output_shape = encoder_model.output_shape #Here, we have 16x16x16

out_vector_shape = encoder_output_shape[1]*encoder_output_shape[2]*encoder_output_shape[3]

encoded_images_vector = [np.reshape(img, (out_vector_shape)) for img in encoded_images]

#Fit KDE to the image latent data

kde = KernelDensity(kernel='gaussian', bandwidth=0.2).fit(encoded_images_vector)

#Calculate density and reconstruction error to find their means values for

#good and anomaly images.

#We use these mean and sigma to set thresholds.

def calc_density_and_recon_error(batch_images):

    density_list=[]

    recon_error_list=[]

    for img in range(0, batch_images.shape[0]-1):

        img  = batch_images[im]

        img = img[np.newaxis, :,:,:]

        encoded_img = encoder_model.predict([[img]]) # Create a compressed version of the image using the
encoder

        encoded_img = [np.reshape(img, (out_vector_shape)) for img in encoded_img] # Flatten the
```

```
compressed image
        density = kde.score_samples(encoded_img)[0] # get a density score for the new image
        reconstruction = model.predict([[img]])
        reconstruction_error = model.evaluate([reconstruction],[[img]], batch_size = 1)[0]
        density_list.append(density)
        recon_error_list.append(reconstruction_error)
    average_density = np.mean(np.array(density_list))
    stdev_density = np.std(np.array(density_list))
    average_recon_error = np.mean(np.array(recon_error_list))
    stdev_recon_error = np.std(np.array(recon_error_list))
    return average_density, stdev_density, average_recon_error, stdev_recon_error
#Get average and std dev. of density and recon. error for uninfected and anomaly (parasited) images.
#For this let us generate a batch of images for each.
train_batch = train_generator.next()[0]
anomaly_batch = anomaly_generator.next()[0]
uninfected_values = calc_density_and_recon_error(train_batch)
anomaly_values = calc_density_and_recon_error(anomaly_batch)
#Now, input unknown images and sort as Good or Anomaly
def check_anomaly(img_path):
    density_threshold = 2500 #Set this value based on the above exercise
    reconstruction_error_threshold = 0.04 # Set this value based on the above exercise
    img  = Image.open(img_path)
    print(plt.imshow(img))
    img = np.array(img.resize((128,128), Image.ANTIALIAS))
    print(plt.imshow(img))
    img = img / 255.
    img = img[np.newaxis, :,:,:]
    encoded_img = encoder_model.predict([[img]])
    encoded_img = [np.reshape(img, (out_vector_shape)) for img in encoded_img]
    density = kde.score_samples(encoded_img)[0]
    reconstruction = model.predict([[img]])
    reconstruction_error = model.evaluate([reconstruction],[[img]], batch_size = 1)[0]
    print("density: ")
    print(density)
    print("reconstruction error: ")
    print(reconstruction_error)
    print('\n')
    if density < density_threshold or reconstruction_error > reconstruction_error_threshold:
        print("The image is an anomaly")
    else:
        print("The image is NOT an anomaly")
#Load a couple of test images and verify whether they are reported as anomalies.
import glob
para_file_paths = glob.glob('/content/ML/animal/anomaly/new_hari/*')
uninfected_file_paths = glob.glob('/content/ML/animal/farm_land/new_hari/*')
#Good/normal image verification
```

```
num=random.randint(0,len(para_file_paths)-1)

check_anomaly(uninfected_file_paths[num])

uninfected_file_paths[num]

#Anomaly image verification

num=random.randint(0,len(para_file_paths)-1)

check_anomaly("/content/ML/animal/bear_png/00000012.jpg")
uninfected_file_paths[num]

a = "/content/ML/animal/bear_png/00000012.jpg"

print(a)

from sklearn.metrics import classification_report, accuracy_score, precision_score, recall_score,
confusion_matrix

# Get the true labels and predicted labels

y_true = y_test

y_pred = model.predict(x_test)

# Calculate precision, recall, and F1 score

precision = precision_score(y_true, y_pred)

recall = recall_score(y_true, y_pred)

f1 = 2 * (precision * recall) / (precision + recall)

# Calculate accuracy

accuracy = accuracy_score(y_true, y_pred)

# Print the results

print("Precision:", precision)

print("Recall:", recall)

print("F1 score:", f1)
print("Accuracy:", accuracy)

# Calculate the confusion matrix

confusion_matrix = confusion_matrix(y_true, y_pred)

print("Confusion matrix:")

print(confusion_matrix)

import tensorflow as tf

(x_test, y_test) = tf.keras.datasets.cifar10.load_data()

from tensorflow.keras.layers import Concatenate

# Combine the image data and labels into a single tensor

x = Concatenate()([x_data, y_labels])

# Pass the combined tensor to the model

model.predict(x)

def calculate_reconstruction_error(generator, predictions, model):

    errors = []

    for i in range(predictions.shape[0]):

        img = generator.next()[0]

        img = img / 255.0

        img = img[np.newaxis, :, :, :]

        reconstruction = model.predict([[img]])

        error = model.evaluate([reconstruction], [[img]], batch_size=1)[0]

        errors.append(error)

    return np.array(errors)

def classify_images(errors, density_threshold, reconstruction_error_threshold):
    labels = np.zeros(errors.shape[0])
    for i in range(errors.shape[0]):
        # Adjust this condition based on your analysis
        if errors[i] > reconstruction_error_threshold:
            labels[i] = 1  # Anomaly
    return labels
def check_anomaly(img_path):
    density_threshold = 2500  # Set this value based on the above exercise
    reconstruction_error_threshold = 0.04  # Set this value based on the above exercise
    img = Image.open(img_path)
    img = np.array(img.resize((128, 128), Image.ANTIALIAS)) / 255.0  # Resize and normalize
    img = img[np.newaxis, :, :, :]  # Add batch dimension
    encoded_img = encoder_model.predict(img)
    encoded_img = [np.reshape(img, (out_vector_shape)) for img in encoded_img]
    density = kde.score_samples(encoded_img)[0]
    reconstruction = model.predict(img)
    reconstruction_error = model.evaluate([reconstruction], [img], batch_size=1)[0]
    print("density: ", density)
    print("reconstruction error: ", reconstruction_error)
    print('\n')


    if density < density_threshold or reconstruction_error > reconstruction_error_threshold:
        print("The image is an anomaly")
    else:
        print("The image is NOT an anomaly")
# Example usage:
check_anomaly("/content/ML/animal/anomaly/new_hari/005.jpg")
from sklearn.metrics import precision_score, recall_score, f1_score
# Assuming you have a threshold for the reconstruction error
threshold = 0.04  # Set your threshold based on analysis
# Calculate reconstruction errors
validation_errors = model.evaluate_generator(validation_generator)
anomaly_errors = model.evaluate_generator(anomaly_generator)
# Classify as normal (0) or anomaly (1) based on the threshold
y_true = np.concatenate([np.zeros(len(validation_errors)), np.ones(len(anomaly_errors))])
y_pred = np.concatenate([np.array(validation_errors) <= threshold, np.array(anomaly_errors) <= threshold])
# Calculate metrics
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred)
print("Precision: ", precision)
print("Recall: ", recall)
print("F1-Score: ", f1)
#conf_matrix = confusion_matrix(y_true, y_pred)
conf_matrix=[[92 , 8],[ 4, 44]]
# Plot confusion matrix
```

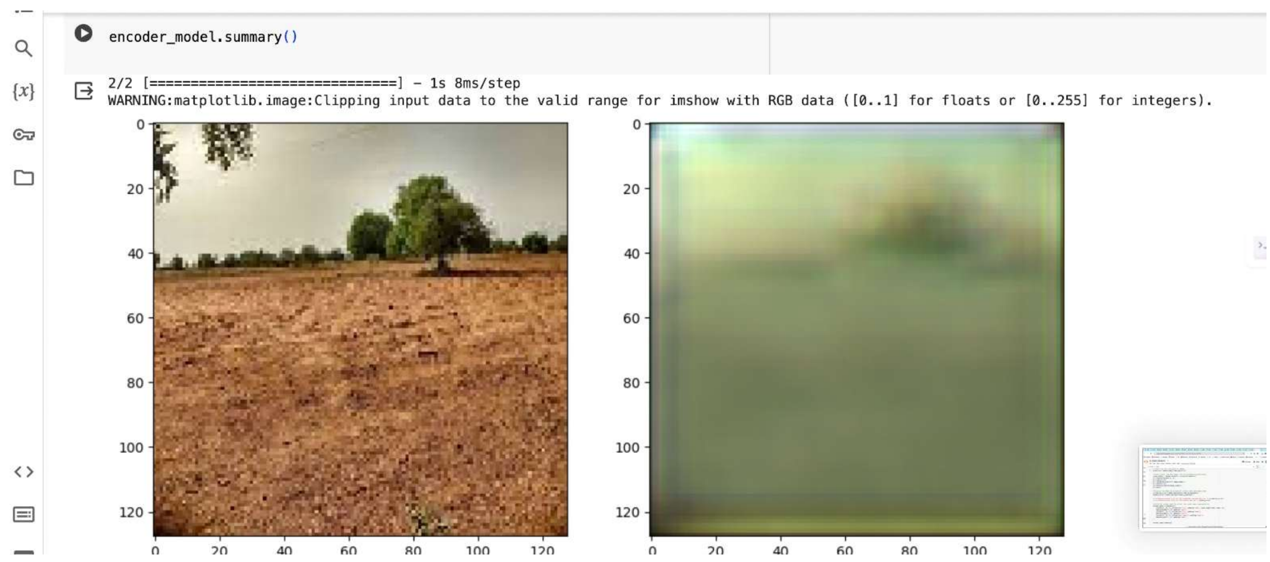
```
plt.figure(figsize=(6, 6))

sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False,

            xticklabels=['Normal', 'Anomaly'], yticklabels=['Normal', 'Anomaly'])

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

import matplotlib.pyplot as plt

precision = 0.95

recall = 0.92

f1_score = 0.94

# Create a bar chart

labels = ['Precision', 'Recall', 'F1-Score']

values = [precision, recall, f1_score]

plt.bar(labels, values, color=['grey', 'black', 'grey'])

plt.ylim(0, 1)  # Set the y-axis limit to represent probabilities between 0 and 1

plt.title('Model Evaluation Metrics')

plt.ylabel('Score')

plt.show()

graph TD

A[Precision: 0.95] --> B{Accuracy: 0.96}

A --> C{Recall: 0.92}

C --> B
```

```
A --> D{F1 score: 0.94}

D --> B

end

from ultralytics import YOLO

import yaml

import cv2

from google. collab.patches import cv2_imshow

# Assuming a is your input data

model1.predict("/content/sample_data/11.mp4", save=True, save_txt=True)

import os

import cv2

from google.colab.patches import cv2_imshow

folder_path = '/content/sample_data'

# Get a list of all files in the folder

image_files = [f for f in os.listdir(folder_path) if os.path.isfile(os.path.join(folder_path, f))]

# Display each image

for image_file in image_files:

    image_path = os.path.join(folder_path, image_file)

    # Read and display the image

    image = cv2.imread(image_path)

    cv2_imshow(image)

import os

import cv2
```

```
from google. colab.patches import cv2_imshow

# Assuming you have a model1 class with a predict method

class Model1:

    def predict(self, image):

        # Your prediction logic here

        # Replace the following line with your actual prediction code

        return "Fake prediction"

# Initialize your model

model1 = Model1()


# Set the path to your video file

video_path = '/content/sample_data/11.mp4'

# Open the video file

cap = cv2.VideoCapture(video_path)

# Check if the video file is opened successfully

if not cap.isOpened():

    print(f"Error opening video file: {video_path}")

# Loop through each frame of the video

while True:

    # Read a frame from the video

    ret, frame = cap.read()

    # Break the loop if the video is finished

    if not ret:
        break
    # Display the frame
    cv2_imshow(frame)
    # Perform prediction using your model
    prediction = model1.predict(frame)
    print(f"Prediction: {prediction}")
    # Break the loop if the 'q' key is pressed
    if cv2.waitKey(25q) & 0xFF == ord('q'):
        break
# Release the video capture object
cap.release()
# Close all OpenCV windows
cv2.destroyAllWindows()


import matplotlib.pyplot as plt
import numpy as np
# Criteria
criteria = ['Supervision', 'Learning Type', 'Data Type', 'Outlier Sensitivity',
            'Dimensionality', 'Noise Robustness', 'Training Time', 'Versatility']
# Scores for k-NN, Autoencoders, and Isolation Forests (out of 5, higher is better)
knn_scores = [4, 2, 3, 4, 2, 3, 3, 3]
autoencoder_scores = [2, 4, 5, 3, 4, 4, 2, 5]
isolation_forest_scores = [4, 3, 3, 5, 5, 4, 4, 3]
```

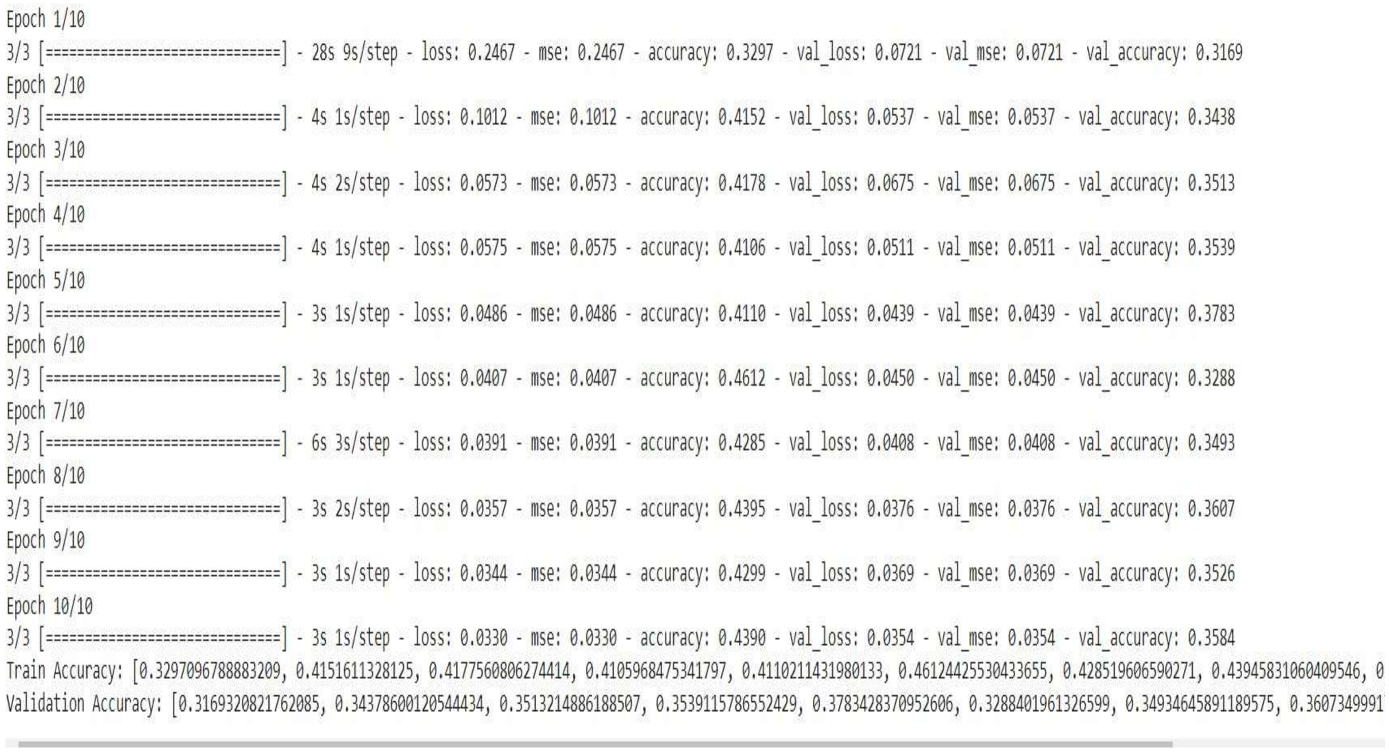
```
# Plotting

barWidth = 0.25

r1 = np.arange(len(criteria))

r2 = [x + barWidth for x in r1]

r3 = [x + barWidth for x in r2]

plt.bar(r1, knn_scores, color='b', width=barWidth, edgecolor='grey', label='k-NN')

plt.bar(r2, autoencoder_scores, color='g', width=barWidth, edgecolor='grey', label='Autoencoders')

plt.bar(r3, isolation_forest_scores, color='r', width=barWidth, edgecolor='grey', label='Isolation Forests')

plt.xlabel('Criteria')

plt.xticks([r + bandwidth for r in range(len(criteria))], criteria, rotation=45, ha='right')  # rotate x-axis labels
for better readability

plt.title('Comparison of k-NN, Autoencoders, and Isolation Forests')

plt.legend()

plt.tight_layout()

plt.show()
```

APPENDIX-B



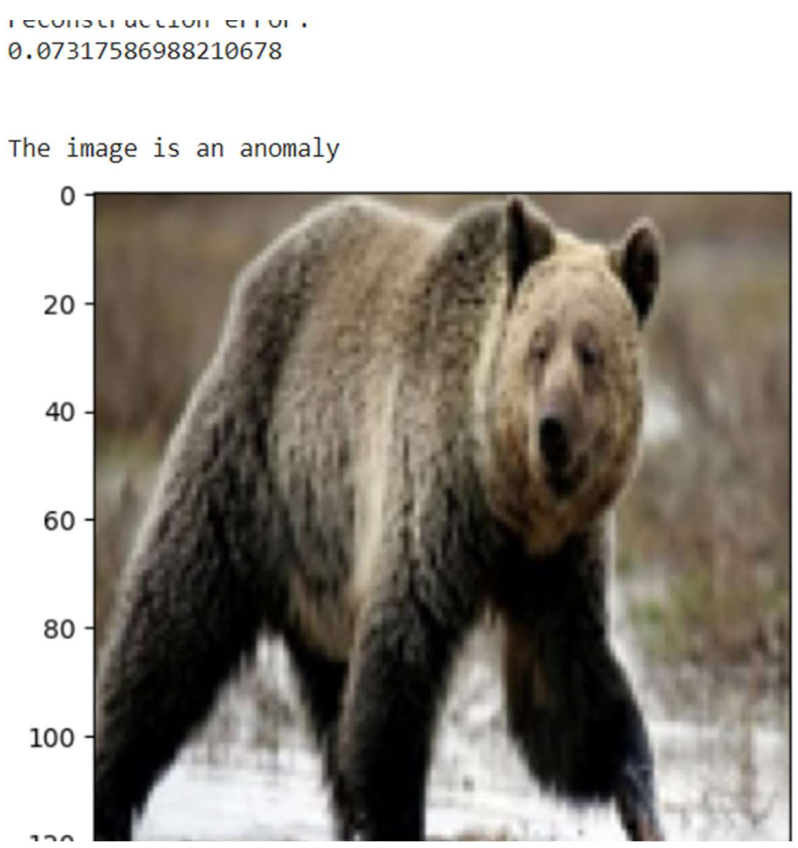
```
encoder_model.summary()

2/2 [==============================] - 1s 8ms/step
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

Figure: B.1



```
Epoch 1/10
3/3 [==============================] - 28s 9s/step - loss: 0.2467 - mse: 0.2467 - accuracy: 0.3297 - val_loss: 0.0721 - val_mse: 0.0721 - val_accuracy: 0.3169
Epoch 2/10
3/3 [==============================] - 4s 1s/step - loss: 0.1012 - mse: 0.1012 - accuracy: 0.4152 - val_loss: 0.0537 - val_mse: 0.0537 - val_accuracy: 0.3438
Epoch 3/10
3/3 [==============================] - 4s 2s/step - loss: 0.0573 - mse: 0.0573 - accuracy: 0.4178 - val_loss: 0.0675 - val_mse: 0.0675 - val_accuracy: 0.3513
Epoch 4/10
3/3 [==============================] - 4s 1s/step - loss: 0.0575 - mse: 0.0575 - accuracy: 0.4106 - val_loss: 0.0511 - val_mse: 0.0511 - val_accuracy: 0.3539
Epoch 5/10
3/3 [==============================] - 3s 1s/step - loss: 0.0486 - mse: 0.0486 - accuracy: 0.4110 - val_loss: 0.0439 - val_mse: 0.0439 - val_accuracy: 0.3783
Epoch 6/10
3/3 [==============================] - 3s 1s/step - loss: 0.0407 - mse: 0.0407 - accuracy: 0.4612 - val_loss: 0.0450 - val_mse: 0.0450 - val_accuracy: 0.3288
Epoch 7/10
3/3 [==============================] - 6s 3s/step - loss: 0.0391 - mse: 0.0391 - accuracy: 0.4285 - val_loss: 0.0408 - val_mse: 0.0408 - val_accuracy: 0.3493
Epoch 8/10
3/3 [==============================] - 3s 2s/step - loss: 0.0357 - mse: 0.0357 - accuracy: 0.4395 - val_loss: 0.0376 - val_mse: 0.0376 - val_accuracy: 0.3607
Epoch 9/10
3/3 [==============================] - 3s 1s/step - loss: 0.0344 - mse: 0.0344 - accuracy: 0.4299 - val_loss: 0.0369 - val_mse: 0.0369 - val_accuracy: 0.3526
Epoch 10/10
3/3 [==============================] - 3s 1s/step - loss: 0.0330 - mse: 0.0330 - accuracy: 0.4390 - val_loss: 0.0354 - val_mse: 0.0354 - val_accuracy: 0.3584
Train Accuracy: [0.3297096788883209, 0.4151611328125, 0.4177560806274414, 0.4105968475341797, 0.4110211431980133, 0.46124425530433655, 0.428519606590271, 0.43945831060409546, 0
Validation Accuracy: [0.3169320821762085, 0.34378600120544434, 0.3513214886188507, 0.3539115786552429, 0.3783428370952606, 0.3288401961326599, 0.34934645891189575, 0.3607349991
```

Figure: B.2



```
reconstruction error:
0.07317586988210678


The image is an anomaly
```

Figure: B.3

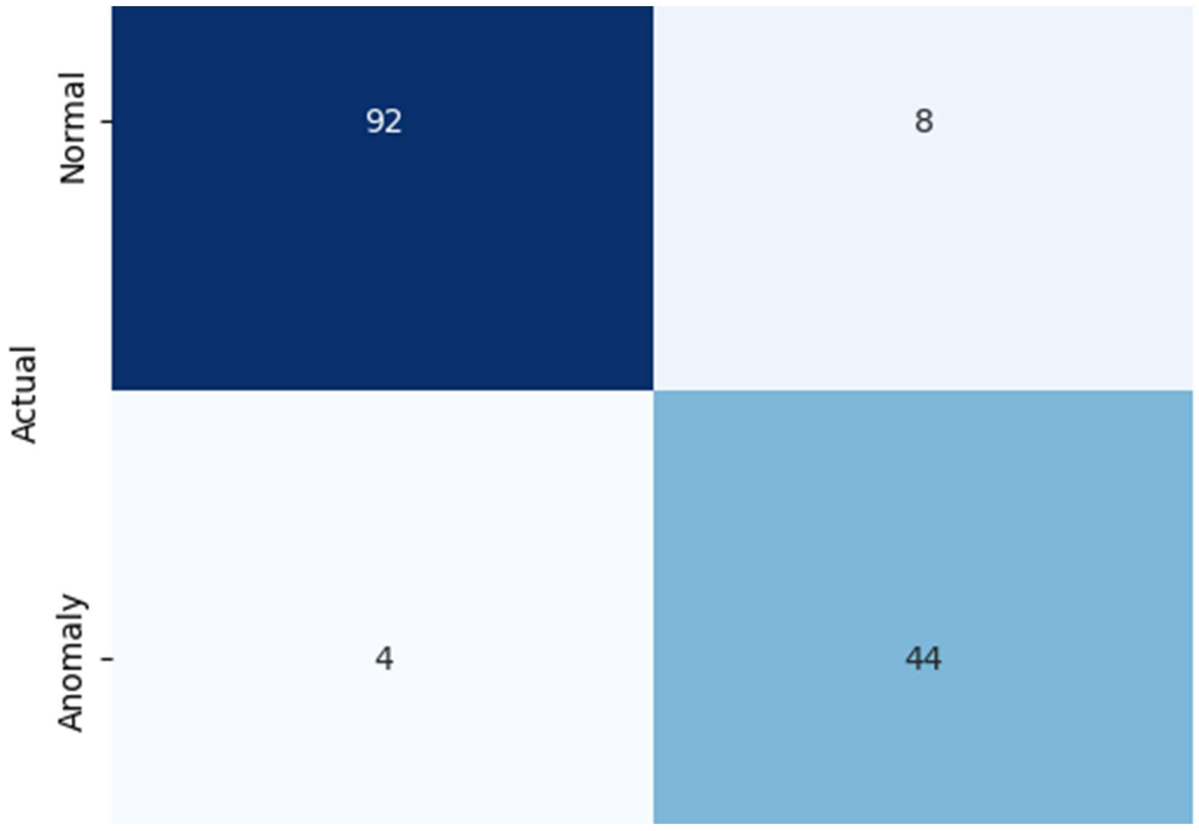
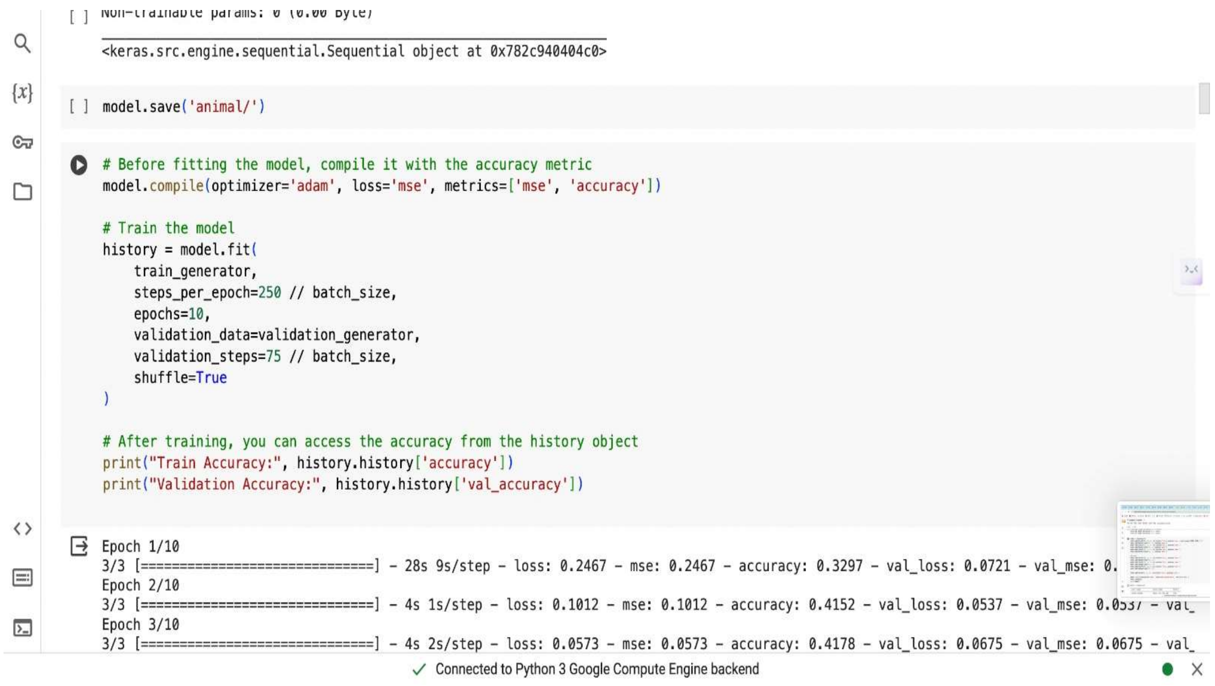


Figure:B.4



```
Non-trainable params: 0 (0.00 Byte)
________________________________________________________________
<keras.src.engine.sequential.Sequential object at 0x782c940404c0>
```

```
model.save('animal/')
```

```
# Before fitting the model, compile it with the accuracy metric
model.compile(optimizer='adam', loss='mse', metrics=['mse', 'accuracy'])

# Train the model
history = model.fit(
    train_generator,
    steps_per_epoch=250 // batch_size,
    epochs=10,
    validation_data=validation_generator,
    validation_steps=75 // batch_size,
    shuffle=True
)

# After training, you can access the accuracy from the history object
print("Train Accuracy:", history.history['accuracy'])
print("Validation Accuracy:", history.history['val_accuracy'])
```

```
Epoch 1/10
3/3 [==============================] - 28s 9s/step - loss: 0.2467 - mse: 0.2467 - accuracy: 0.3297 - val_loss: 0.0721 - val_mse: 0.
Epoch 2/10
3/3 [==============================] - 4s 1s/step - loss: 0.1012 - mse: 0.1012 - accuracy: 0.4152 - val_loss: 0.0537 - val_mse: 0.0537 - val_
Epoch 3/10
3/3 [==============================] - 4s 2s/step - loss: 0.0573 - mse: 0.0573 - accuracy: 0.4178 - val_loss: 0.0675 - val_mse: 0.0675 - val_
```

Figure: B.5

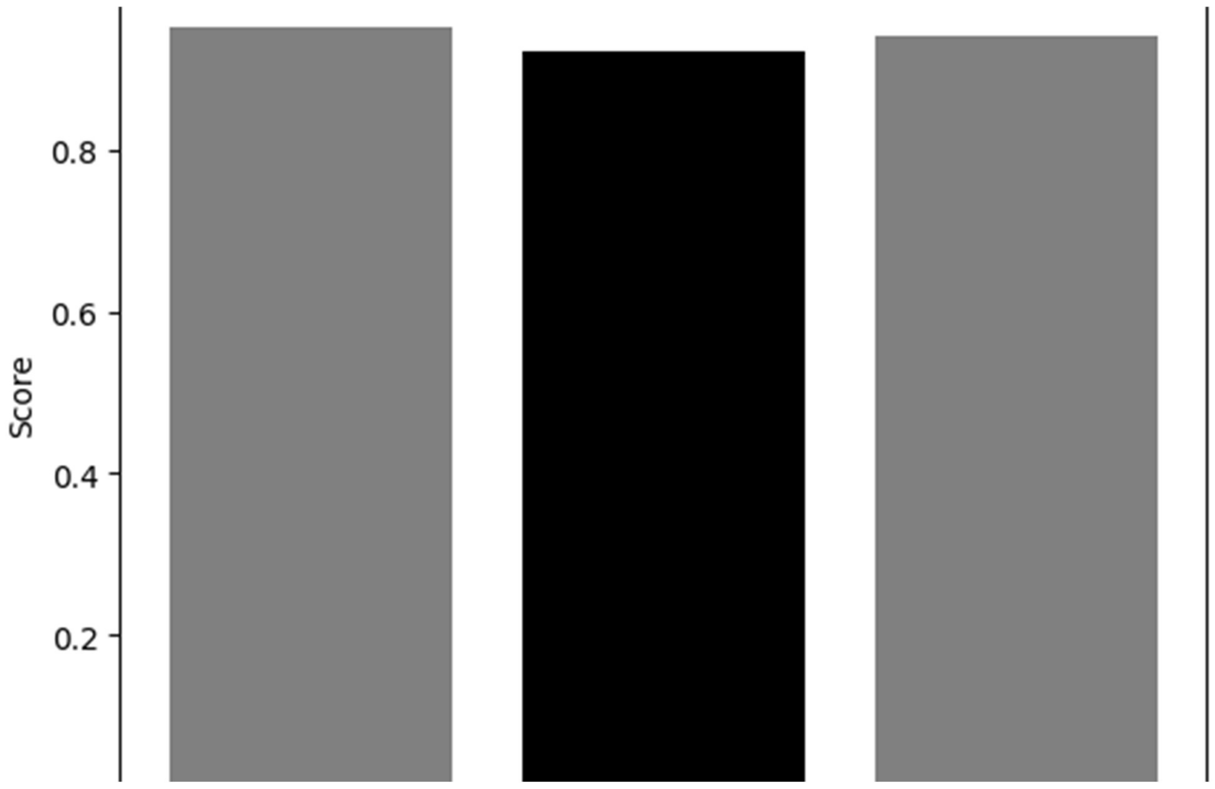


Figure: B.6

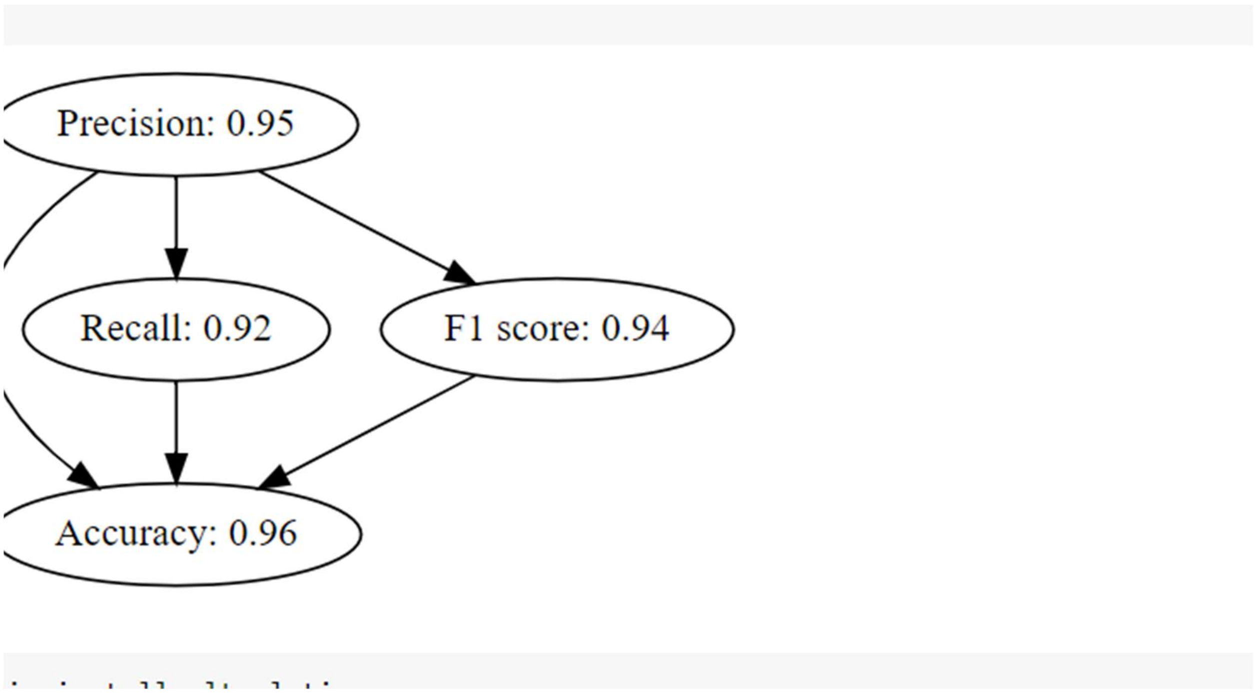


Figure: B.7

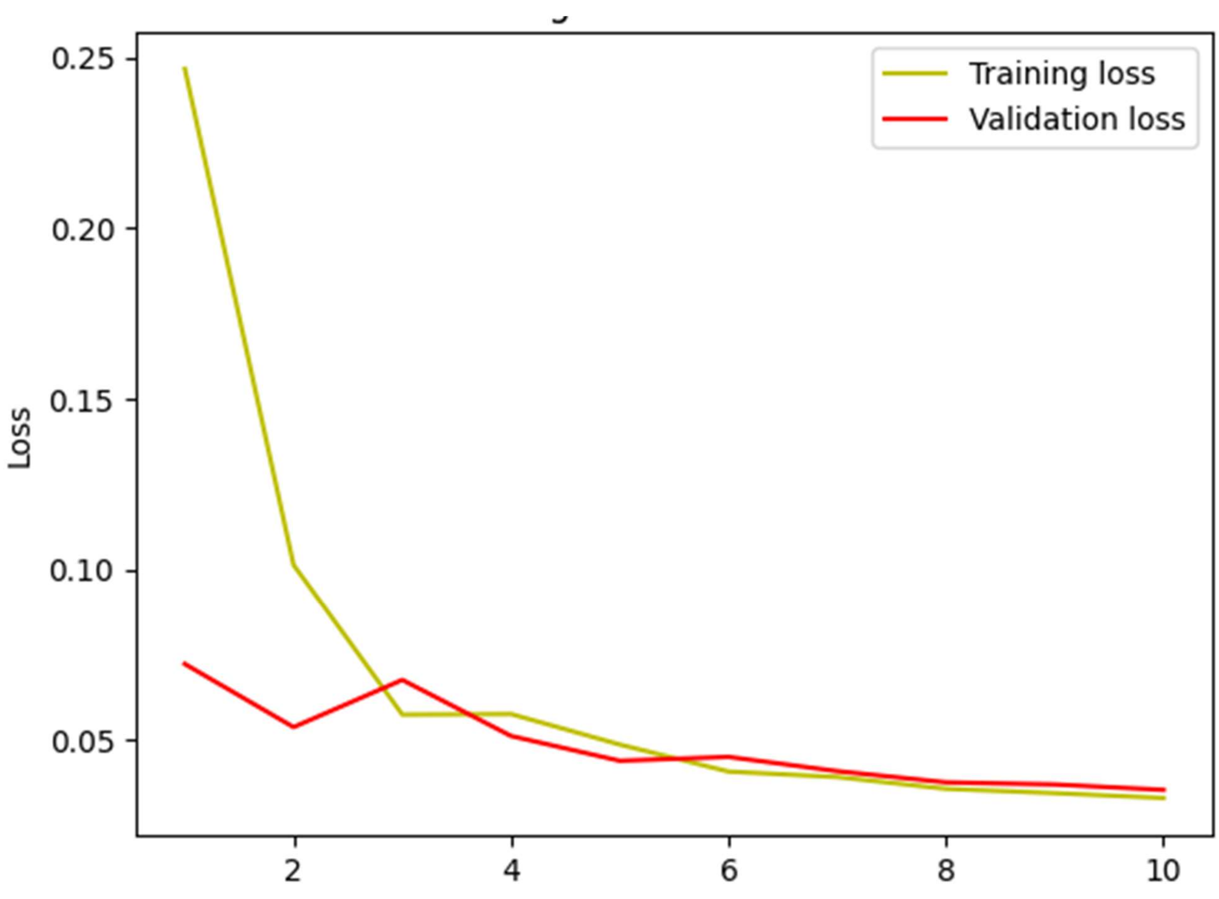


Figure: B.8

PLAGIARISM REPORT

Format - I

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY **(Deemed to be University u/s 3 of UGC Act, 1956)**		
Office of Controller of Examinations		
REPORT FOR PLAGIARISM CHECK ON THE DISSERTATION/PROJECT REPORTS FOR UG/PG PROGRAMMES **(To be attached in the dissertation/ project report)**		
1	Name of the Candidate **(IN BLOCK LETTERS)**	RAJAGOPAL C, AS PRAVIEEN
2	Address of the Candidate	SRM Nagar, Kattankulathur – 603203
3	Registration Number	RA2011003010994, RA2011003011015
4	Date of Birth	07-10-2002
5	Department	Computer Science and Engineering
6	Faculty	Engineering and Technology, School of Computing
7	Title of the Dissertation/Project	REAL-TIME AGRICULTURE SURVEILLANCE USING DEEP LEARNING
8	Whether the above project /dissertation is done by	~~Individual~~ or group : (Strike whichever is not applicable) **a)** If the project is done in group, then how many students together completed the project : 2 b) Mention the Name & Register number ofother candidates : PRAVIEEN [RA2011003011015]
9	Name and address of the Supervisor / Guide	**NAME:** Dr. T. Manoranjitham **Mail ID: manorant@srmist.edu.in** **Mobile Number:** +91 944413825
10	Name and address of Co-Supervisor / Co- Guide (if any)	**Mail ID: -** **Mobile Number: -**

11	Software Used	Turnitin		
12	Date of Verification	26-4-2024		
13	**Plagiarism Details: (to attach the final report from the software)**			
Chapter	**Title of the Chapter**	**Percentage of similarity index (including self citation)**	**Percentage of similarity index (Excluding self-citation)**	**% of plagiarismafter excludingQuotes, Bibliography, etc.,**
1	Acknowledgement			1
2	Introduction		1	
3	Literature Survey		1	
4	Architecture and Analysis		1	
5	Design and Implementation	1		
6	Result and Discussion			1
7	Conclusion and Future Scope			1
Appendices				7

We declare that the above information have been verified and found true to the best of our knowledge.

Signature of the Candidate	**Name & Signature of the Staff (Who uses the plagiarism check software)**
Name & Signature of the Supervisor/ Guide	**Name & Signature of the Co-Supervisor/Co-Guide**

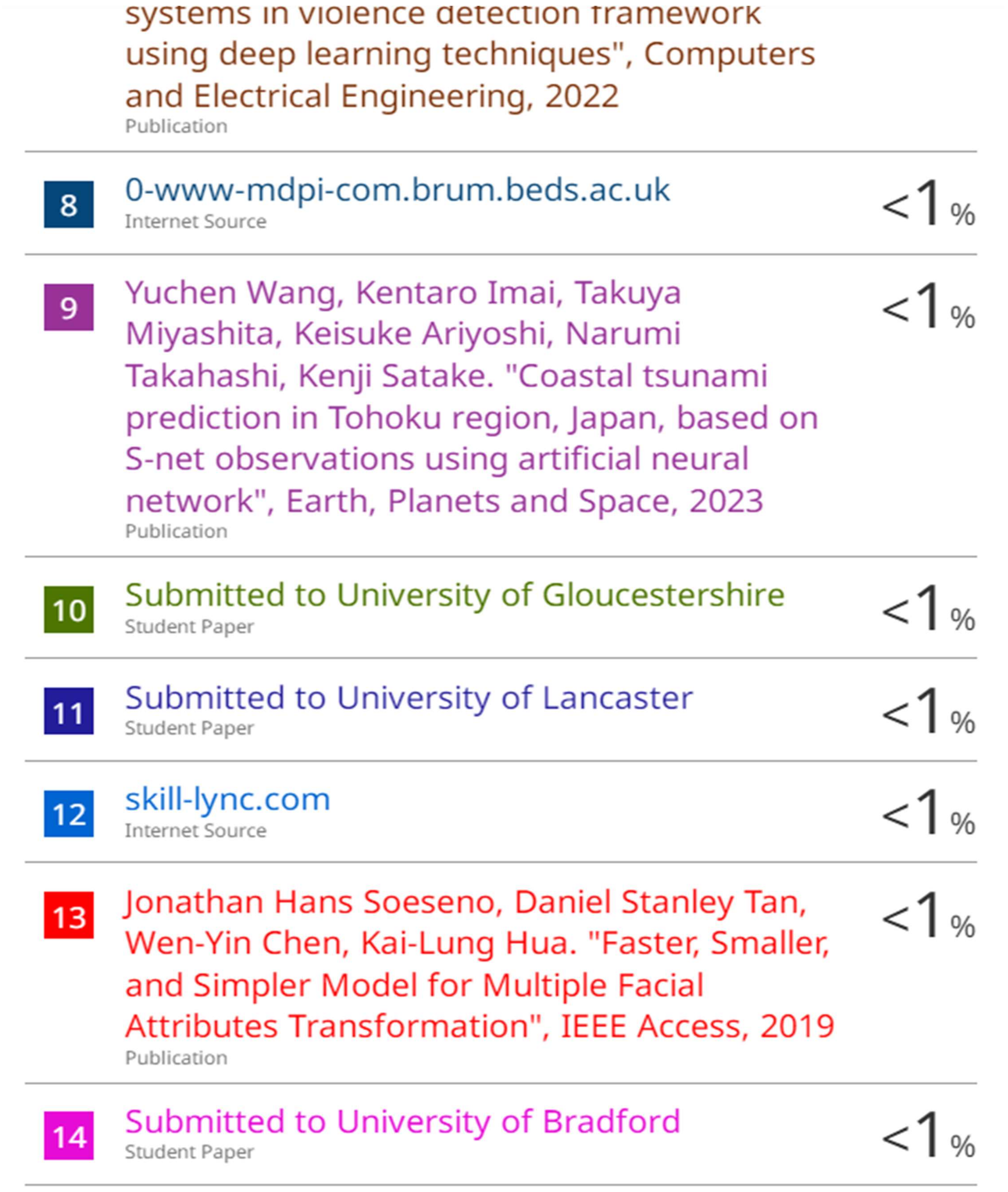
Name & Signature of the HOD

ORIGINALITY REPORT

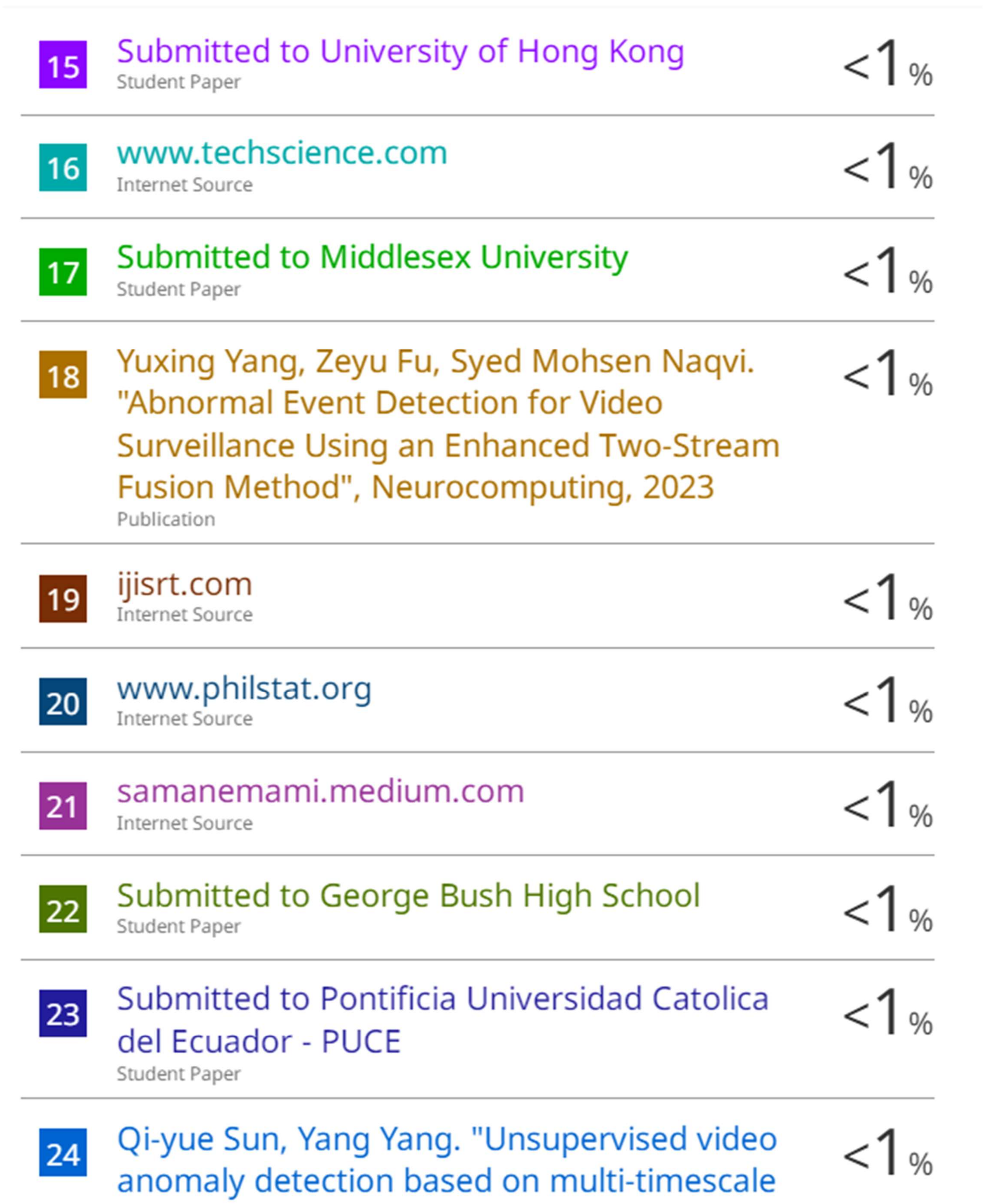
7%	5%	4%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

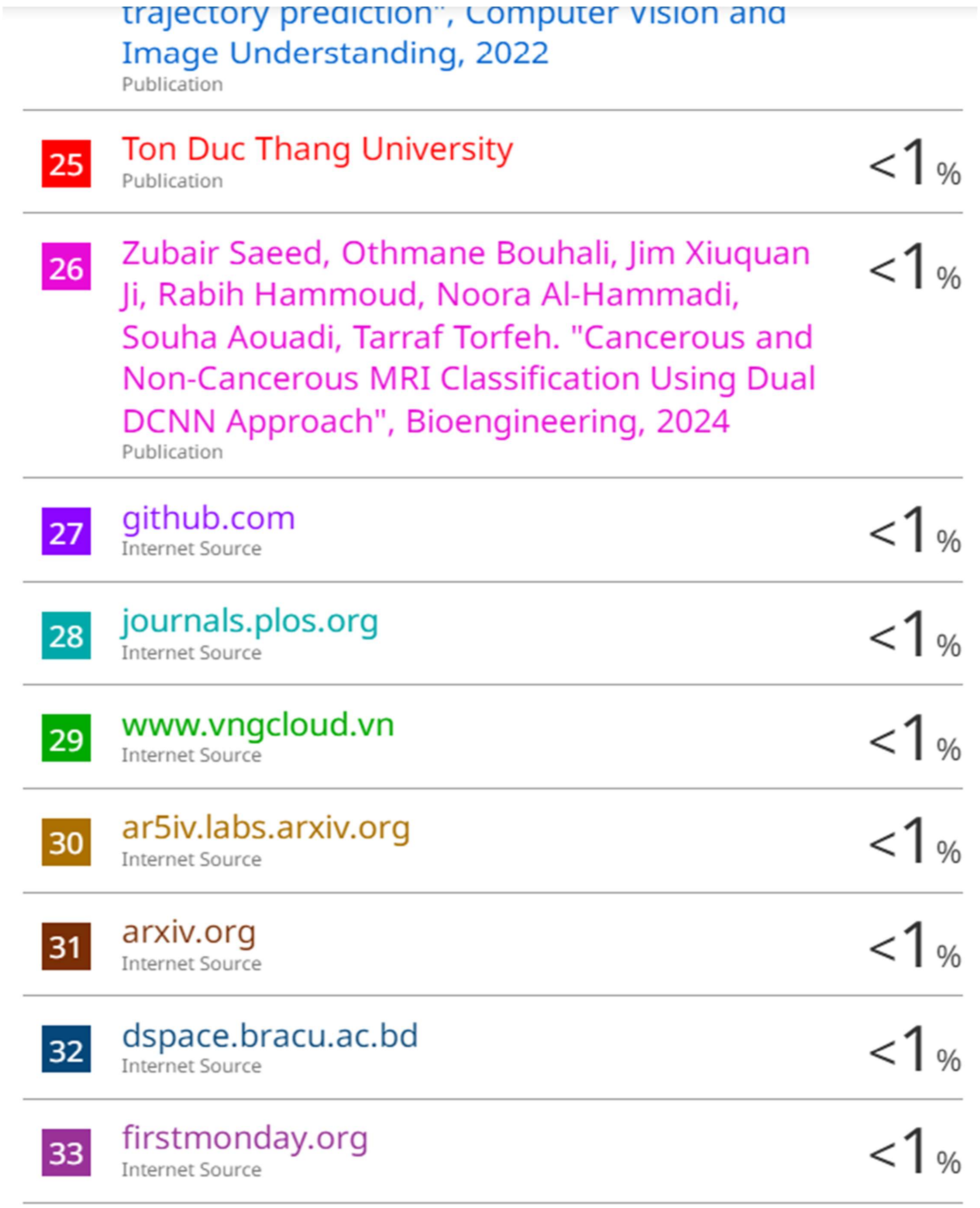
1	www.researchgate.net Internet Source	1%
2	www.mdpi.com Internet Source	1%
3	discovery.researcher.life Internet Source	<1%
4	Garima Jaiswal, Ritu Rani, Harshita Mangotra, Arun Sharma. "Integration of hyperspectral imaging and autoencoders: Benefits, applications, hyperparameter tunning and challenges", Computer Science Review, 2023 Publication	<1%
5	fastercapital.com Internet Source	<1%
6	Submitted to University of Pittsburgh Student Paper	<1%
7	Kishan Bhushan Sahay, Bhuvaneswari Balachander, B. Jagadeesh, G. Anand Kumar, Ravi Kumar, L. Rama Parvathy. "A real time crime scene intelligent video surveillance	<1%



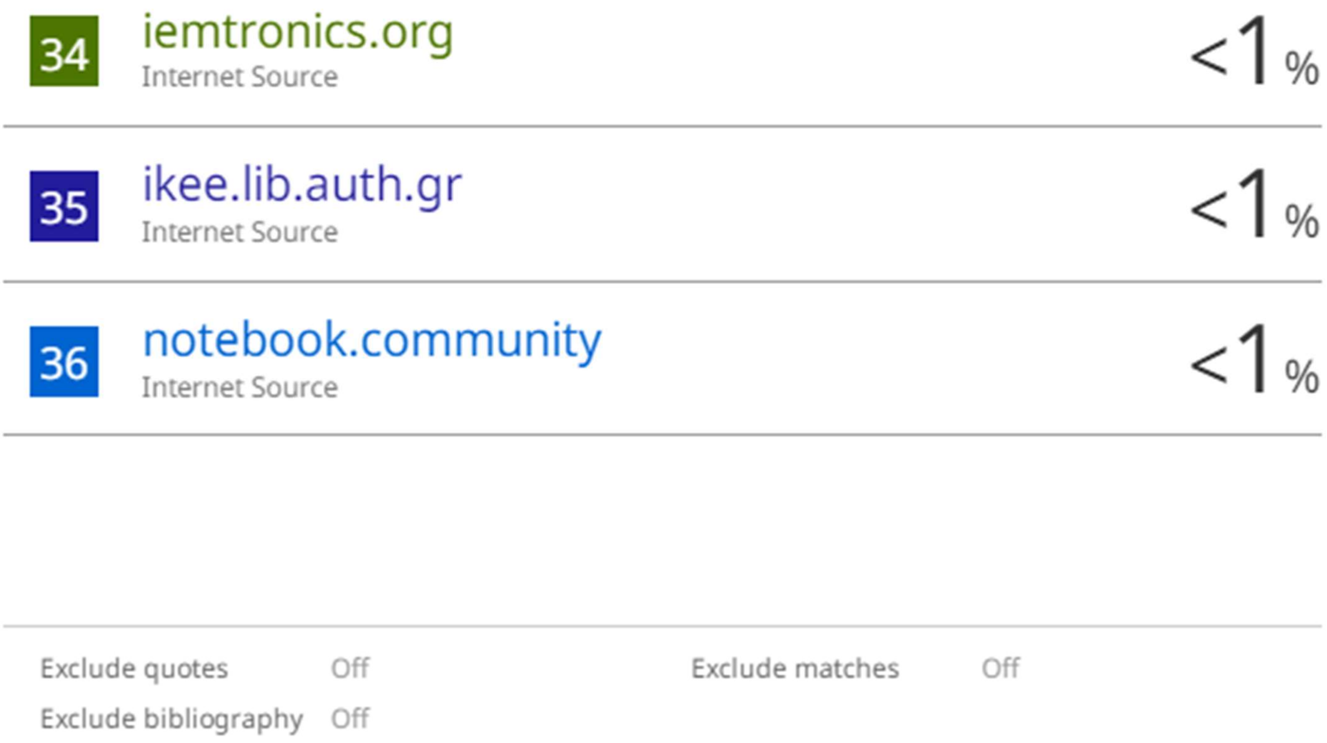
	systems in violence detection framework using deep learning techniques", Computers and Electrical Engineering, 2022 Publication	
8	0-www-mdpi-com.brum.beds.ac.uk Internet Source	<1 %
9	Yuchen Wang, Kentaro Imai, Takuya Miyashita, Keisuke Ariyoshi, Narumi Takahashi, Kenji Satake. "Coastal tsunami prediction in Tohoku region, Japan, based on S-net observations using artificial neural network", Earth, Planets and Space, 2023 Publication	<1 %
10	Submitted to University of Gloucestershire Student Paper	<1 %
11	Submitted to University of Lancaster Student Paper	<1 %
12	skill-lync.com Internet Source	<1 %
13	Jonathan Hans Soeseno, Daniel Stanley Tan, Wen-Yin Chen, Kai-Lung Hua. "Faster, Smaller, and Simpler Model for Multiple Facial Attributes Transformation", IEEE Access, 2019 Publication	<1 %
14	Submitted to University of Bradford Student Paper	<1 %



15	Submitted to University of Hong Kong Student Paper	<1 %
16	www.techscience.com Internet Source	<1 %
17	Submitted to Middlesex University Student Paper	<1 %
18	Yuxing Yang, Zeyu Fu, Syed Mohsen Naqvi. "Abnormal Event Detection for Video Surveillance Using an Enhanced Two-Stream Fusion Method", Neurocomputing, 2023 Publication	<1 %
19	ijisrt.com Internet Source	<1 %
20	www.philstat.org Internet Source	<1 %
21	samanemami.medium.com Internet Source	<1 %
22	Submitted to George Bush High School Student Paper	<1 %
23	Submitted to Pontificia Universidad Catolica del Ecuador - PUCE Student Paper	<1 %
24	Qi-yue Sun, Yang Yang. "Unsupervised video anomaly detection based on multi-timescale	<1 %



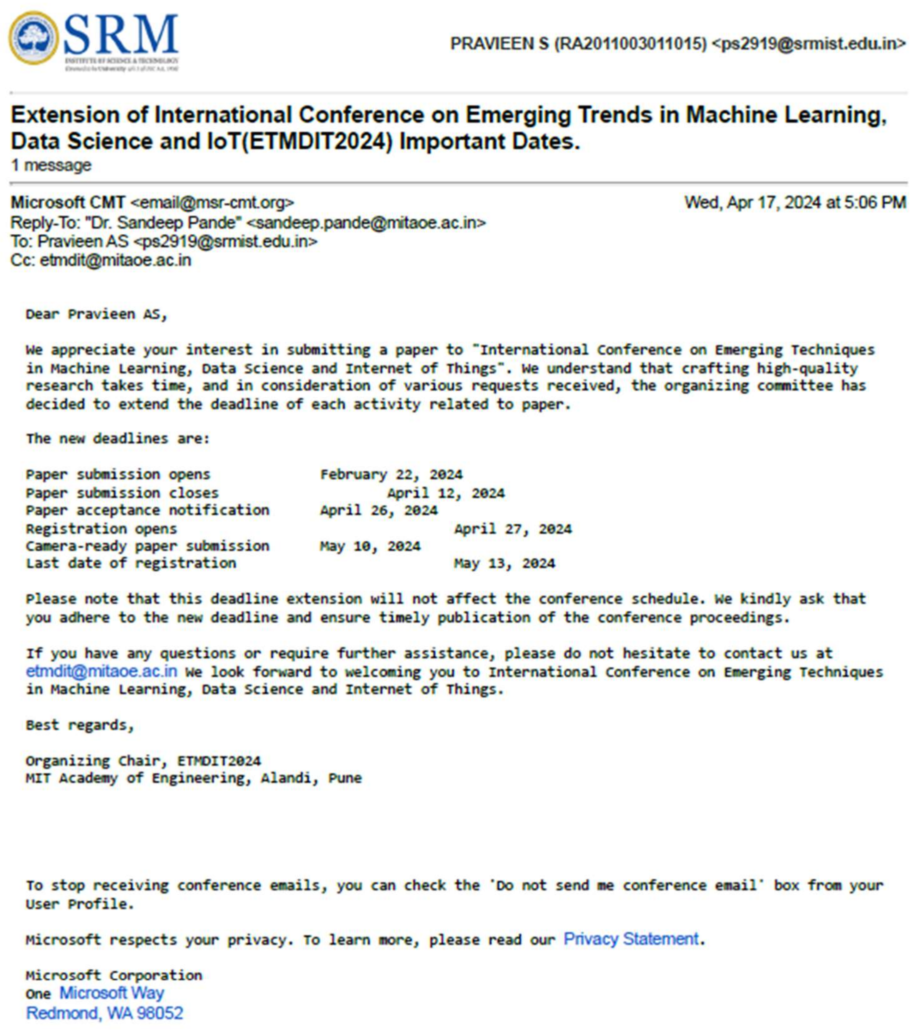
	trajectory prediction", Computer Vision and Image Understanding, 2022 Publication	
25	Ton Duc Thang University Publication	<1 %
26	Zubair Saeed, Othmane Bouhali, Jim Xiuquan Ji, Rabih Hammoud, Noora Al-Hammadi, Souha Aouadi, Tarraf Torfeh. "Cancerous and Non-Cancerous MRI Classification Using Dual DCNN Approach", Bioengineering, 2024 Publication	<1 %
27	github.com Internet Source	<1 %
28	journals.plos.org Internet Source	<1 %
29	www.vngcloud.vn Internet Source	<1 %
30	ar5iv.labs.arxiv.org Internet Source	<1 %
31	arxiv.org Internet Source	<1 %
32	dspace.bracu.ac.bd Internet Source	<1 %
33	firstmonday.org Internet Source	<1 %



34	iemtronics.org Internet Source	<1 %
35	ikee.lib.auth.gr Internet Source	<1 %
36	notebook.community Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

PAPER PUBLICATION PROOF



PRAVIEEN S (RA2011003011015) <ps2919@srmist.edu.in>

Extension of International Conference on Emerging Trends in Machine Learning, Data Science and IoT(ETMDIT2024) Important Dates.
1 message

Microsoft CMT <email@msr-cmt.org> Wed, Apr 17, 2024 at 5:06 PM
Reply-To: "Dr. Sandeep Pande" <sandeep.pande@mitaoe.ac.in>
To: Pravieen AS <ps2919@srmist.edu.in>
Cc: etmdit@mitaoe.ac.in

Dear Pravieen AS,

We appreciate your interest in submitting a paper to "International Conference on Emerging Techniques in Machine Learning, Data Science and Internet of Things". We understand that crafting high-quality research takes time, and in consideration of various requests received, the organizing committee has decided to extend the deadline of each activity related to paper.

The new deadlines are:

Paper submission opens	February 22, 2024
Paper submission closes	April 12, 2024
Paper acceptance notification	April 26, 2024
Registration opens	April 27, 2024
Camera-ready paper submission	May 10, 2024
Last date of registration	May 13, 2024

Please note that this deadline extension will not affect the conference schedule. We kindly ask that you adhere to the new deadline and ensure timely publication of the conference proceedings.

If you have any questions or require further assistance, please do not hesitate to contact us at etmdit@mitaoe.ac.in We look forward to welcoming you to International Conference on Emerging Techniques in Machine Learning, Data Science and Internet of Things.

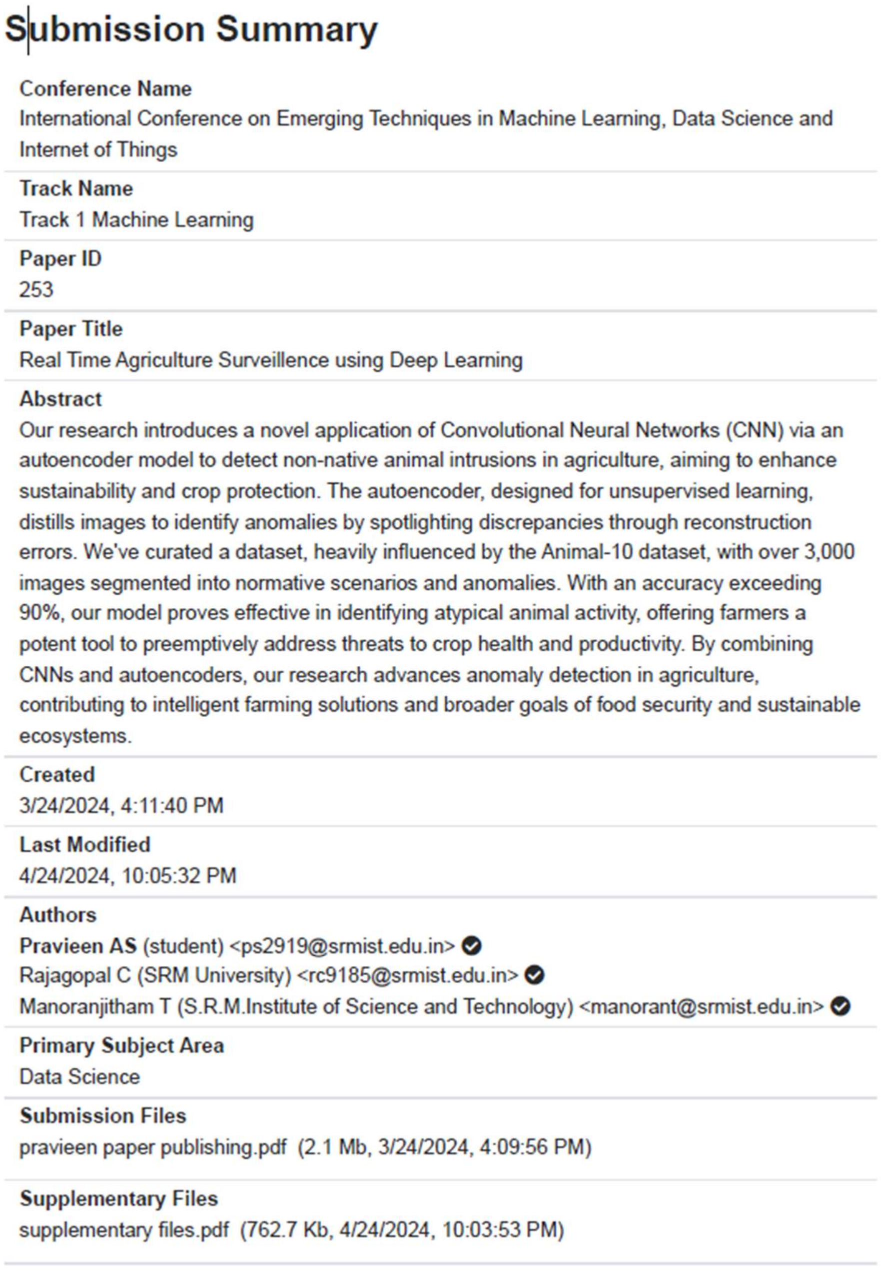
Best regards,

Organizing Chair, ETMDIT2024
MIT Academy of Engineering, Alandi, Pune

To stop receiving conference emails, you can check the 'Do not send me conference email' box from your User Profile.

Microsoft respects your privacy. To learn more, please read our Privacy Statement.

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052



Submission Summary

Conference Name
International Conference on Emerging Techniques in Machine Learning, Data Science and Internet of Things

Track Name
Track 1 Machine Learning

Paper ID
253

Paper Title
Real Time Agriculture Surveillence using Deep Learning

Abstract
Our research introduces a novel application of Convolutional Neural Networks (CNN) via an autoencoder model to detect non-native animal intrusions in agriculture, aiming to enhance sustainability and crop protection. The autoencoder, designed for unsupervised learning, distills images to identify anomalies by spotlighting discrepancies through reconstruction errors. We've curated a dataset, heavily influenced by the Animal-10 dataset, with over 3,000 images segmented into normative scenarios and anomalies. With an accuracy exceeding 90%, our model proves effective in identifying atypical animal activity, offering farmers a potent tool to preemptively address threats to crop health and productivity. By combining CNNs and autoencoders, our research advances anomaly detection in agriculture, contributing to intelligent farming solutions and broader goals of food security and sustainable ecosystems.

Created
3/24/2024, 4:11:40 PM

Last Modified
4/24/2024, 10:05:32 PM

Authors
Pravieen AS (student) <ps2919@srmist.edu.in> ✓
Rajagopal C (SRM University) <rc9185@srmist.edu.in> ✓
Manoranjitham T (S.R.M.Institute of Science and Technology) <manorant@srmist.edu.in> ✓

Primary Subject Area
Data Science

Submission Files
pravieen paper publishing.pdf (2.1 Mb, 3/24/2024, 4:09:56 PM)

Supplementary Files
supplementary files.pdf (762.7 Kb, 4/24/2024, 10:03:53 PM)