BORDER DEFENCE MECHANISM CLASSIFICATIONUSING DEEP LEARNING TECHNIQUES

A PROJECT REPORT

Submitted by

LEELA S	**(211420205081)**
RITHIKA RANJITH VP	**(211420205126)**
SHALINI M	**(211420205139)**

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MARCH 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonmous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"BORDER DEFENCE MECHANISM CLASSIFICATION USING DEEP LEARNING TECHNIQUES"** is the bonafide work of **"Leela. S (211420205081), Rithika Ranjith.V.P (211420205126), Shalini. M(211420205139)"** who carried out the project under my supervision.

SIGNATURE	**SIGNATURE**
Dr. M. HELDA MERCY M.E., Ph.D.,	**Mrs. J. HEMAVATHY**
HEAD OF THE DEPARTMENT	**B.TECH., M.E., (Ph. D).,**
	ASSISTANT PROFESSOR
	SUPERVISOR
Department of Information Technology	Department of Information Technology
Panimalar Engineering College	Panimalar Engineering College
Poonamallee, Chennai - 600 123	Poonamallee, Chennai - 600 123

Submitted for the project and viva-voce examination held on__________________________.

SIGNATURE	**SIGNATURE**
INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DECLARATION

We hereby declare that the project report entitled "**BORDER DEFENCE MECHANISM CLASSIFICATION USING DEEP LEARNING TECHNIQUES** " which is being submitted in partial fulfilment of the requirement of the course leading to the award of the 'Bachelor Of Technology in Information Technology ' in **Panimalar Engineering College, Autonomous institution Affiliated to Anna university- Chennai** is the result of the project carried out by us under the guidance of **Mrs. J. HEMAVATHY,B.TECH , M.E (Ph.D.)., ASSISTANT PROFESSOR in the Department of Information Technology**. We further declared that I or any other person has not previously submitted this project report to any other institution/university for any other degree/ diploma or any other person.

LEELA. S (211420205081)

RITHIKA RANJITH. V.P (211420205126)

SHALINI. M (211420205139)

Date:

Place**:** Chennai

It is certified that this project has been prepared and submitted under my guidance.

Date: Mrs. J. HEMAVATHY

Place: Chennai **(ASSISTANT PROFESSOR / IT)**

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co-operation and support from many, for successful completion . We wish to express our sincere thanks to all those who were involved in the completion of this project.

Our sincere thanks to **Our Beloved Secretary and Correspondent,Dr. P. CHINNADURAI, M.A., Ph.D.,** for his sincere endeavor in educating us in his premier institution.

We would like to express our deep gratitude to **Our Dynamic Directors, Mrs. C. VIJAYA RAJESHWARI and Dr. C. SAKTHI KUMAR, M.E.,M.B.A.,Ph.D.,** and **Dr. SARANYA SREE SAKTHIKUMAR.,B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities for completion of this project.

We also express our appreciation and gratefulness to **Our Principal Dr. K. MANI, M.E., Ph.D.,** who helped us in the completion of the project. We wish to convey our thanks and gratitude to our head of the department, **Dr. M. HELDA MERCY, M.E., Ph.D.,** Department of Information Technology, for her support and by providing us ample time to complete our project.

We express our indebtedness and gratitude to our Project co-ordinator **Mr. M. DILLI BABU**, **M.E.,(P.hD.,)** Associate Professor, Department of Information Technology for his guidance throughout the course of our project. We also express sincere thanks to our supervisor **Mrs. J. HEMAVATHY,B.TECH , M.E (Ph.D.).,** Assistant Professor, Department of Information Technology for providing the support to carry out the project successfully. Last, we thank our parents and friends for providing their extensive moral support and encouragement during the course ofthe project.

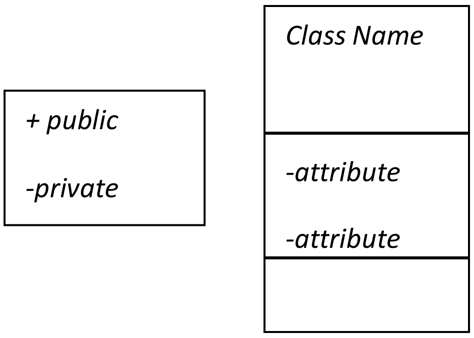
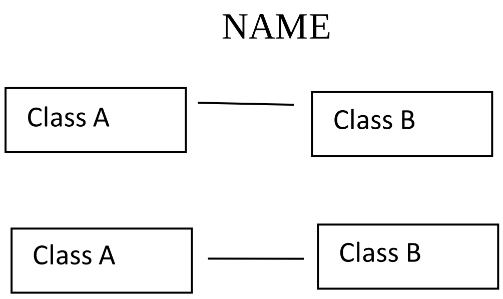
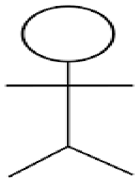
TABLE OF CONTENT

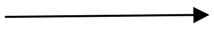
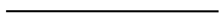
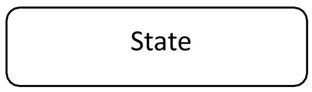
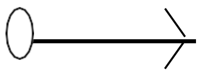
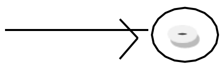
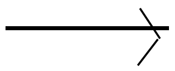
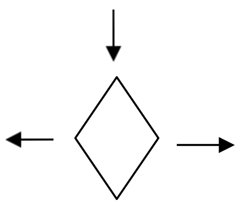
ABSTRACT

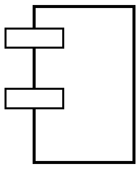
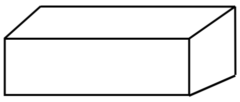
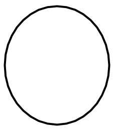
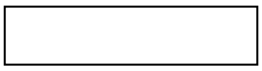
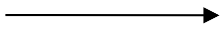
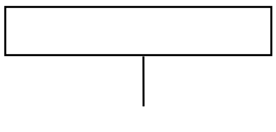
In the domain of national security and border protection, the development of effective defense mechanisms holds paramount significance. Traditional strategies often hinge on human intervention and manual surveillance, which are resource-intensive and susceptible to humanerror. This paper introduces a pioneering approach to border defense mechanism classification employing deep learning techniques. The primary objective is to design and implement an intelligent system capable of automatically classifying and identifying various border defense mechanisms, such as fences, walls, trenches, and sensor-based systems, from a diverse array of data sources including images and sensor data. This system leverages Convolutional Neural Networks (CNNs) for spatial feature extraction and Recurrent Neural Networks (RNNs) for capturing temporal patterns, constituting multiple stages of development. Initially, a large and meticulously curated dataset encompassing images and sensor data of diverse border defense mechanisms will be amassed for training the deep learning models. Optimization techniques will be employed to fine-tune model parameters, ensuring the attainment of optimal performance levels. Upon completion of the training process, the models will be seamlessly integrated into a unified system capable of processing real-time data streams emanating from an array of surveillance sources such as cameras, satellite imagery, and ground sensors. This integrated system is engineered to bolster situational awareness, facilitate prompt threat detection, and empower data-driven decision-making processes in border security operations. Emphasizing adaptability and interoperability, this novel approach endeavors to fortify national security endeavors by furnishing an intelligent and comprehensive solution to the multifaceted challenges posed by border protection requirements. In conclusion, this innovative deep learning-based approach represents a significant advancement in border security and defense mechanism classification. Emphasizing adaptability and interoperability, this approach is poised to strengthen national security efforts by providing an intelligent and comprehensive solution to the complex challenges of border protection.

LIST OF FIGURES

LIST OF SYSMBOLS

S.NO	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class	 + public -private Class Name -attribute -attribute	Represents a collection of similar entities grouped together.
2.	Association		Associations represents static relationships between classes. Roles represents the way the two classes see each other.
3.	Actor		It aggregates several classes into a single classes.

4.	*Relation* (uses)	*uses*	Used for additional process communication.
5.	Relation (extends)	EXTENDS 	Extends relationship is used when one use case is similar to another use case but does a bit more.
6.	Communicatio n		Communication between various use cases.
7.	State		State of the process.
8.	Initial State		Initial state of the object
9.	Final state		Final state of the object
10.	Control flow		Represents various control flow between the states.
11.	Decision box		Represents decision making process from a constraint

12.	Component		Represents physical modules which is a collection of components.
13.	Node		Represents physical modules which are a collection of components.
14.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or acion.
15.	External entity		Represents external entities such as keyboard,sensors,etc.
16.	Transition		Represents communication that occurs between processes.
17.	Object Lifeline		Represents the vertical dimensions that the object communications.

LIST OF ABBREVATIONS

CNN	Convolutional Neural Networks
HD	High Definition
ML	Machine Learning
AI	Artificial Intelligence
ATM	Air Traffic Management
CAD	Computer-Aided Design
RGB	Red Green Blue
GPU	Graphics Processing Unit
URL	Uniform Resource Locator
API	Application Programming Interface
ILSVRC	ImageNet Large Scale Visual Recognition Challenge

LIST OF TABLES

CHAPTER – 1
INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

The project centers on leveraging cutting-edge image processing technologies, particularly high-resolution imagery and advanced machine learning algorithms, to bolster security measures within the aviation and transportation sectors. The primary focus areas are border defense and passenger aircraft classification, where precise object detection is indispensable for maintaining safety and security protocols.

High-resolution imagery forms the cornerstone of the project, offering unparalleled clarity and detail compared to traditional image sources. By harnessing the enhanced resolution provided by modern imaging systems such as high-definition cameras and satellite imagery, the project aims to achieve a granular level of object detection accuracy. High-resolution images facilitate the identification and localization of target objects with unprecedented precision, crucial for tasks like monitoring borders or identifying aircraft in congested airspace.

Convolutional Neural Networks (CNNs) emerge as the cornerstone of the project's machine learning framework for object detection. CNNs have demonstrated exceptional prowess in image recognition tasks, thanks to their ability to learn complex hierarchical features directly from raw pixel data. In the context of border defense and aircraft classification, CNNs excel in discerning intricate patterns and features amidst diverse backgrounds and environmental conditions. By training CNN models on extensive datasets comprising diverse examples of target objects, the project endeavors to develop robust and highly accurate object detection systems. A key emphasis of the project lies in addressing the unique challenges posed by the complex operational environments inherent to border defense and passenger aircraft classification.

These scenarios often involve cluttered backgrounds, varying lighting conditions, and the presence of multiple object classes. To navigate these challenges, the project adopts a multi-faceted approach,incorporating techniques such as data augmentation, transfer learning, and ensemble modeling to enhance the robustness and adaptability of the object detection system.

Real-time or near real-time object detection capabilities are essential for ensuring timely responses to security threats or operational anomalies. To this end, the project places a strong emphasis on optimizing the computational efficiency of the CNN-based object detection models. Through model optimization techniques such as network pruning, quantization, and hardware acceleration, the project aims to achieve high-speed inference without compromising detection accuracy.

Automation is a central theme throughout the project, aimed at reducing reliance on manual surveillance methods and streamlining security operations. By integrating CNN-based object detection systems with existing surveillance infrastructure, the project seeks to automate the detection and tracking of target objects, thereby enhancing situational awareness and response capabilities.

By developing robust, efficient, and automated object detection systems tailored to the specific requirements of border defense and aircraft classification, the project endeavors to enhance safety, security, and operational efficiency across diverse operational domains. The project aims to enhance safety and security by developing efficient, automated object detection systems tailored for border defense and aircraft classification. Additionally, we're focusing on optimizing these systems to minimize false positives and maximize resource allocation efficiency. Moreover, we're prioritizing real-time data fusion techniques to provide comprehensive situational awareness and enable proactive threat response capabilities.

1.2 MOTIVATION:

The identification and classification of border defense and passenger aircraft types hold significant importance in various domains, including national security and civilian aviation. With a vast number of these aircraft actively operating worldwide, accurately determining their types poses a considerable challenge. Manual identification methods are often time- consuming, error-prone, and labor-intensive, particularly in situations where rapid decision- making is crucial. Additionally, the diversity of aircraft models, configurations, and camouflage techniques further complicates the classification process. Hence, there is a pressing need for automated systems capable of swiftly and accurately identifying border defense and passenger aircraft types. By leveraging Convolutional Neural Networks (CNNs),this project aims to address these challenges and provide a robust solution for automated aircraft type classification. Such a system would enhance situational awareness, streamline security operations, and ultimately contribute to bolstering national security and ensuring thesafety and efficiency of civilian air travel worldwide.

1.3 OBJECTIVE OF THE WORK:

In response to the imperative need for border defense image classification, an extensive approach will be undertaken. Leveraging Convolutional Neural Networks (CNNs), a sophisticated deep learning model will be meticulously developed. This endeavor involves a comprehensive comparative analysis of diverse CNN architectures to discern the most precise and efficient model. With a focus on precision and robustness, the chosen CNN model will be meticulously trained on a curated dataset specifically tailored to simulate real-world border defense scenarios. In the pursuit of optimal performance, an intricate combination of hyperparameter tuning and data augmentation techniques will be employed to refine the model. Subsequent validation of the trained model's effectiveness will be conducted rigorouslyusing a dedicated test dataset. The culmination of this rigorous process will be the delivery ofinsightful deployment recommendations, ensuring seamless integration and efficacy of the model in actual border defense operations, thereby fortifying security measures effectively.

1.4 PROBLEM STATEMENT

In the expansive domains of aviation and transportation security, the imperative to ensure the integrity of borders and accurately classify passenger aircraft stands as a cornerstone of safeguarding national and international interests. However, the landscape of conventional surveillance techniques often reveals inadequacies in meeting the escalating demands of detecting potential threats with the requisite accuracy and efficiency. This glaring disparity underscores the pressing need for the development of a robust deep learning model, leveraging the sophi sticated capabilities of Convolutional Neural Networks (CNNs), specifically tailored for border defense image classification. This multifaceted endeavor encompasses several pivotal components,commencing with the meticulous selection of the most suitable CNN architecture, characterized by its ability to effectively capture intricate spatial features inherent in border defense imagery. Subsequently, the identified CNN model embarks on a journey of rigorous training, utilizing meticulously curated border defense datasets that encapsulate the complexities an d nuances of real-world scenarios. Optimization of the model's performance constitutes a crucial phase, necessitating comprehensive exploration of hyperparameter tuning strategies and judiciousapplication of data augmentation techniques to enhance the model's capacity to generalize and accurately classify a diverse array of border defense imagery. Following the exhaustive trainingand optimization regimen, the efficacy and reliability of the developed model will be subjected to rigorous validation using a meticulously curated test dataset, providing empirical evidence of its prowess in accurately identifying and categorizing objects of interest within border d efense scenarios. The overarching objective of this ambitious project is to engender an innovative solution that substantially enhances security protocols by furnishing precise identification and categorization capabilities within the complex milieu of border defense operations. Through this concerted endeavor,we aspire to make a significant contribution to the advancement of safety measures in aviation and transportation, thereby fortifying the resilience of critical infrastructure against potential threats and ensuring the safety and security of individuals and assets on a global scale. In summary, the development of a robust deep learning model tailored for border defense image classification using Convolutional Neural Networks (CNNs) holds promise for enhancing aviation and transportation security. Through meticulous selection of architectures, rigorous training, and optimization, our model aims to accurately detect threats along borders.

CHAPTER 2
LITERATURE SURVEY

2.1 LITERATURE SURVEY

(1) An Adaptive Framework for Optimization and Prediction of AirTraffic Management (Sub-)Systems with Machine Learning

Author : Stefan Reitmann ,and Michael Schultz
Year : 2022

The framework outlined in this paper stands as a monumental guidepost in the labyrinthine landscape of air traffic management (ATM) evaluation and optimization. By meticulously weaving together principles of mathematical modeling and machine learning, it furnishes a systematic and exhaustive methodology that transcends the conventional boundaries of analysis. At its core, this framework serves as a beacon of enlightenment, illuminating the darkened corridors where time-continuous system dynamics intersect with the discrete tapestry of measured data and performance metrics. Its conceptual scaffolding, carefully erected upon the pillars of abstraction and formalization, enables a panoramic vista of ATM operations, unshackled from the fetters of direct observation. Through the judicious utilization of representative datasets, meticulously curated from the annals of past experiments and publications, the framework metamorphoses into a potent prognosticator, capable of divining the trajectory of boarding times and unraveling the enigma of flight delays. While the intricate intricacies of neural network training are expounded upon in separate treatises, this paper embarks on a voyage of exploration into the vast expanse of control mechanisms and optimization strategies, sowed from the fertile soil of these trained models. In summation, this framework transcends mere methodology; it stands as a colossus, an indomitable edifice upon which the future of ATM efficiency and reliability is inscribed. It empowers stakeholders with the alchemical elixir of knowledge, guiding their hand as they navigate the tempestuous seas of air traffic management, forging a path toward safer skies and smoother journeys for all who traverse them

(2) A Multi-Step CNN-Based Estimation of Aircraft Landing Gear Angles
Author: Fuyang Li , Zhiguo Wu
Year : 2021

This paper presents a method for measuring aircraft landing gear angles based on a monocular camera and the CAD aircraft model. Condition monitoring of the aircraft landing gear is a prerequisite for the safe landing of the aircraft. Traditional manual observation has an intense subjectivity. In recent years, target detection models dependent on deep learning and pose estimation methods relying on a single RGB image have made significant progress. Based on these advanced algorithms, this paper proposes a method for measuring the actual angles of landing gears in twodimensional images. A single RGB image of an aircraft is inputted to the target detection module to obtain the key points of landing gears. The vector field network votes the key points of the fuselage afterextraction and scale normalization of the pixels inside the aircraft prediction box.Knowing the pixel position of the key points and the constraints on the aircraft, the angle between the landing gear and fuselage plane can be calculated even without depth information. The vector field loss function is improved based on the distance between pixels and key points, and synthetic datasets of aircraft withdifferent angle landing gears are created to verify the validity of the proposed algorithm. In summary, this paper introduces a new method for measuring aircraft landing gear angles using a monocular camera and CAD aircraft models. By leveraging advanced target detection and pose estimation techniques, the method accurately calculates landing gear angles from single RGB images. Experimental results validate the algorithm's effectiveness, achieving a mean error of less than 5 degrees across various lighting conditions. This approach holds promise for automated and precise aircraft landing gear monitoring, enhancing aviation safety and efficiency.

(3) Vehicle Detection and Traffic Density Monitoring from Very High Resolution Satellite Video Data

Author: G . Kopsiaftis , k . karantzalos

Year:2015

In this paper an automated vehicle detection and traffic density estimation algorithm has been developed and validated for very high resolution satellite video data. The algorithm is based on an adaptive background estimation procedure followed by a background subtraction at every video frame. The vehicle detection is performed through a further mathematical morphology and statistical analysis on the computed connected components. The traffic density has been estimated based on a lower resolution grid superimposed on the scene. In particular, at every subregion the number of the detected vehicles is calculated and the density is then estimated for the entire road network at every frame. The developed algorithm has been quantitatively evaluated. The quite promising results indicate the potentials of the proposed approach, while parallel GPU implementations can allow for real-time performance.

Moving from single image analysis to video sequence processing, in this paper, the goal was to exploit the recent very high resolution satellite video data that small satellite missions like Skybox Imaging Inc. can deliver In summary, this paper introduces an automated vehicle detection and traffic density estimation algorithm tailored for very high resolution satellite video data. By leveraging adaptive background estimation, background subtraction, and mathematical morphology, the algorithm accurately detects vehicles and estimates traffic density. Quantitative evaluation demonstrates promising results, indicating the algorithm's potential for real-world applications. Additionally, parallel GPU implementations offer the prospect of achieving real-time performance, enhancing its practicality for traffic monitoring using satellite video data.

(4) Scene change detection in the hard disk drive embedded digital satellite receiver for video indexing

Author: Y. K. Seong, Y.-H. Choi, J.-A. Park, and T. S. Choi

Year: 2002

In our paper, we unveil a revolutionary advancement: a hard disk drive embedded digital satellite receiver featuring cutting-edge scene change detection for video indexing. This groundbreaking technology not only facilitates the seamless storage and retrieval of broadcast data but also offers sophisticated classification capabilities. By establishing an interface between conventional digital satellite receivers and digital storage media, our innovative system empowers users with unparalleled access to a vast array of information, significantly enhancing the efficiency of video retrieval processes. This monumental breakthrough represents a paradigm shift in the realm of digital satellite receivers, poised to redefine user experiences and expectations in accessing and managing broadcast content. With this pioneering solution, we anticipate a transformative impact on the landscape of satellite broadcasting, opening up new avenues for enhanced functionality and user convenience on a global scale. In conclusion, our paper presents a pioneering hard disk drive embedded digital satellite receiver with scene change detection for video indexing. This innovative receiver revolutionizes the way broadcast data is stored, retrieved, and classified, bridging the gap between conventional digital satellite receivers and digital storage media. By enabling more efficient video retrieval through advanced scene change detection, users can access a wealth of information with unprecedented ease and accuracy. This represents a significant advancement in digital satellite receiver technology, promising enhanced user experiences and opening up new possibilities for accessing and managing broadcast content. Overall, our solution heralds a new era of functionality and convenience in satellite broadcasting, paving the way for future developments in the field.

(5) A Method to Detect and Track Moving Airplanes from a Satellite Video

Author: Fan Shi 1 , Fang Qiu 1,* , Xiao Li 1 , Yunwei Tang 2 , Ruofei

Year: 2020

In recent years, the deployment of satellites equipped with advanced video capturing capabilities has ushered in a new era of satellite imagery, offering high-definition videos that extend far beyond the traditional scope of remotely sensed data. While the detection and tracking of moving objects are fundamental tasks in this domain, existing research has predominantly concentrated on vehicles, overlooking the intricate challenges posed by other dynamic entities, particularly airplanes. To address this gap, our study endeavors to accurately detect and track airplanes within satellite videos, necessitating a comprehensive approach to overcome several key challenges. Firstly, the presence of slow-moving airplanes can induce foreground aperture problems during detection, necessitating innovative solutions. Secondly, the occurrence of various disturbances, such as parallax motion, presents a formidable obstacle, often leading to false detections. Lastly, the complex maneuvers executed by airplanes demand a tracking algorithm that is both rotation-invariant and scale-invariant to ensure robust performance across diverse scenarios. To surmount these hurdles, we propose an Improved Gaussian-based Background Subtractor (IPGBBS) algorithm tailored specifically for the detection of moving airplanes. Leveraging a novel strategy for background and foreground adaptation, this algorithm effectively mitigates issues associated with foreground aperture, ensuring accurate detection. Subsequently, we employ a Primary Scale Invariant Feature Transform (P-SIFT) keypoint matching algorithm to track detected airplanes. By leveraging P-SIFT keypoints, which exhibit superior distinctiveness and repeatability, our approach enhances the accuracy and reliability of airplane tracking in satellite videos, paving the way for a more comprehensive understanding of aerial dynamics in remote sensing applications.

(6) Remote sensing object tracking with deep reinforcement learning under occlusion

Author: Y. Cui, B. Hou, Q. Wu, B. Ren, S. Wang, and L. JiaoY

Year: 2021

Object tracking in space Earth observation within the realm of remote sensing constitutes a critical research area, pivotal for various applications. Despite the notable successes achieved by existing correlation filter-based and deep learning (DL)-based algorithms, challenges persist, particularly in dealing with object occlusion scenarios. The complex interplay of background changes and tracking lens deviations often results in missing object information and subsequent detection omissions. Traditionally, addressing occlusion involves deploying intricate network models to re-detect occluded objects, which can be computationally intensive. To overcome this challenge, we propose a novel approach. Firstly, we introduce an innovative Action Decision-Occlusion Handling Network (AD-OHNet) based on deep reinforcement learning (DRL). This network is specifically designed to achieve low computational complexity while effectively addressing object tracking under occlusion. Secondly, we leverage temporal and spatial context, along with object appearance models and motion vectors, to provide crucial occlusion information. This information guides actions in reinforcement learning, particularly in scenarios of complete occlusion, thereby enhancing tracking accuracy without compromising on speed. Lastly, we validate the proposed AD-OHNet framework through comprehensive evaluations conducted on three distinct remote sensing video datasets. This novel approach holds promise for advancing the state-of-the-art in object tracking within space Earth observation, offering enhanced accuracy and efficiency in challenging occlusion scenarios.

(7) A powerful random technique to estimate the background in video sequences

Author : O Barnich, M Van Droogenbroeck

Year: 2009

Background subtraction is a critical component in various automatic video content analysis applications. While numerous techniques have been proposed for background extraction, there remains a pressing need for more efficient algorithms that can adapt to diverse environments, exhibit robustness against noise, and maintain high computational efficiency. In this paper, we present a powerful method for background extraction that not only enhances accuracy but also reduces computational burden. Our key innovation lies in the utilization of a random policy for selecting values to construct a samples-based estimation of the background. This marks the first instance of incorporating random aggregation in the realm of background extraction. Furthermore, we introduce a novel policy for information propagation among neighboring pixels in an image. Detailed experiments outlined in this paper demonstrate the superiority of our method over widely used techniques, particularly in scenarios involving noisy images. The random selection policy ensures a smooth exponentially decaying lifespan for sample values, enabling effective handling of concurrent events with a single model of reasonable size for each pixel. This innovative approach promises to advance the state-of-the-art in background subtraction, offering enhanced adaptability, resilience, and computational efficiency for automatic video content analysis. In summary, our paper introduces a novel method for background subtraction in automatic video content analysis. By leveraging a random policy for sample selection and introducing a unique information propagation policy, we achieve improved accuracy and reduced computational load. Overall, our method promises enhanced adaptability, resilience, and computational efficiency for a wide range of video analysis applications

2.2 EXISTING SYSTEM

Utilizing remote sensing video for monitoring aircraft dynamics holds significant implications across various domains, including military applications, airport management, and aircraft rescue operations. Given the fixed size and discernible characteristics of aircraft, correlation filtering emerges as a suitable technique for tracking. Correlation filtering algorithms excel in extracting features from input data to predict motion trajectories swiftly, thereby offering a significant advantage for tracking targets in remote sensing images. In this article, we introduce an antidrift multifilter tracker, amalgamating a correlation filter with the Kalman filter, specifically tailored for this purpose. Our proposed approach, termed the temporal consistency-constrained background-aware correlation filter algorithm, incorporates temporalregularization to combat model drift, particularly induced by cloud occlusion, by leveraging motion information for corrective measures. Through comprehensive experimentation, our proposed method demonstrates superior antidrift performance compared to other advanced tracking methodologies, particularly evident in scenarios with cloud occlusion, while maintaining stable performance in other complex conditions. We envisage that our model willserve as a valuable resource for researchers interested in object tracking in satellite video, especially for effectively processing satellite video data afflicted by cloud occlusion, thereby advancing capabilities in remote sensing and enhancing situational awareness in critical applications.

Drawbacks On Existing System:

- They track only aerial vehicle process.
- Development cost much higher.
- They did not use any deep learning techniques.
- Higher complexity process.

CHAPTER – 3
SYSTEM DESIGN

3.1 PROPOSED SYSTEM

The proposed border defense mechanism classification system represents a groundbreaking initiative aimed at fortifying border security through the sophisticated utilization of advanced deep learning methodologies, prominently featuring convolutional neural networks (CNNs). By integrating data streams from a diverse array of sensors, encompassing cameras, drones, and thermal sensors, the system establishes a comprehensive and resilient surveillance framework, ensuring pervasive coverage along the border. This multifaceted sensor integration not only broadens the scope and depth of monitoring capabilities but also facilitates the capture of nuanced details and diverse environmental conditions, thereby empowering decision-makers with a more nuanced understanding of border activities. At the heart of the system's effectiveness lies its adept use of deep learning algorithms, which endow it with the capacity to autonomously identify potential threats and discern anomalies amidst the multitude of border activities. Leveraging the intrinsic capabilities of CNNs to extract salient features from visual data, the system adeptly discriminates between routine operations and suspicious behavior, enabling border patrol agents to swiftly prioritize and respond to potential security threats with precision and efficiency. An indispensable aspect contributing to the system's efficacy is its adaptive learning capabilities, facilitated by extensive training on meticulously labeled datasets. Through iterative learning processes, the system continuously refines its ability to recognize intricate patterns and subtle deviations, progressively enhancing its accuracy and efficacy in threat detection. This iterative learning paradigm not only facilitates continuous improvement in performance but also ensures the system's resilience and adaptability to evolving threat landscapes and operational exigencies. Moreover, the seamless integration of real-time alerting mechanisms within the system facilitates swift response times, providing border patrol agents with timely notifications upon the detection of suspicious activities. This proactive approach empowers rapid and decisive interventions, thereby bolstering the overall security posture of the border. The system's architectural design is meticulously crafted to accommodate dynamic operational landscapes characterized by evolving threat scenarios and technological advancements. Regular updates and retraining protocols constitute integral components of the system's maintenance strategy, endowing it with the flexibility and resilience necessary to stay abreast of emerging security challenges. In essence, the proposed border defense mechanism classification system embodies a holistic approach to border security, harnessing the power of cutting-edge deep learning methodologies to augment surveillance capabilities, enhance threat detection accuracy, and enable swift response actions. Through its innovative design and adaptive capabilities, the system represents a monumental advancement in the domain of border security, contributing significantly to the overarching goal of safeguarding national borders and protecting critical infrastructure against emerging threats

In conclusion, the proposed border defense mechanism classification system represents a pivotal advancement in border security, offering a sophisticated solution to the complex challenges faced in safeguarding national borders. By harnessing cutting-edge deep learning methodologies, particularly convolutional neural networks (CNNs), and integrating data streams from diverse sensor sources like cameras, drones, and thermal sensors, the system establishes a robust surveillance framework characterized by extensive coverage and unparalleled accuracy. Its adaptive learning capabilities, refined through rigorous training on meticulously labeled datasets, enable continual enhancement of threat detection efficacy by recognizing intricate patterns and subtle deviations. The seamless incorporation of real-time alerting mechanisms ensures swift responses, empowering border patrol agents with timely notifications upon detecting suspicious activities, thereby facilitating proactive interventions to mitigate potential security breaches. Furthermore, the system's meticulously crafted architectural design is tailored to accommodate dynamic operational landscapes inherent in border security, ensuring resilience and adaptability to evolving threat scenarios and technological advancements. In essence, the proposed border defense mechanism classification system represents a monumental stride forward, embodying a proactive and comprehensive approach to enhancing national security and safeguarding critical infrastructure, ultimately fostering the safety and well-being of citizens and assets.

Advantage of proposed system:

- We build own architecture of this project.
- We build a framework based application for deployment purpose.
- We classify ten aerial vehicles.
- We compared more than a two architecture to getting better accuracy level.

3.2 DATASET URL:

3.2.1 DATASET DESCRIPTION:

The dataset employed in this study constitutes a comprehensive collection of image records encompassing extracted features categorized into 20 distinct classes. This dataset is meticulously divided into two subsets: the training data and the test data. The training set assumes a pivotal role as the primary input for model learning, wherein each image record is meticulously associated with a known output label. This structured arrangement facilitates the model's capacity to generalize patterns and features effectively, thereby enhancing its ability to classify unseen data accurately. Conversely, the test dataset serves as a critical tool for evaluating the performance of trained models. Through rigorous assessment, the goal is to ascertain the model's accuracy and effectiveness in classifying images across the predefined categories. To facilitate dataset manipulation, model development, and evaluation processes, this study leverages TensorFlow, a widely acclaimed machine learning library, in conjunction with the Keras method in Python. This powerful combination of tools and frameworks ensures efficient and seamless execution of tasks, facilitating comprehensive analysis and validation of the proposed model's efficacy in image classification tasks.

3.2.2 .MODULE 1: MANUAL NET

This module represents a component where human involvement plays a crucial role alongside automated processes. Its functions encompass various tasks essential for refining and augmenting the system's performance. These tasks include verifying automated classifications to ensure accuracy, correcting any errors or f alse positives, providing decision support in complex scenarios, handling exceptions that fall outside the system's parameters, and annotating training data to continually improve the model's learning capabilities. By integrating human judgment and expertise, the manual module serves to enhance the reliability and effectiveness of the border defense mechanism classification system, contributing to more robust security operations

3.2.3 MODULE 2: VGG NET

A typical deep Convolutional Neural Network (CNN) architecture with several layers is called Visual Geometry Group (abbreviated VGG). Thirteen and sixteen convolutional layers, respectively, comprise VGG-19. Modern The VGG architecture serves as the foundation for object identification models A deep neural network called VGG Net beats baselines outside of ImageNet on a variety of tasks and datasets. It is still one of the most used structures for image recognition today."Elevated Efficiency Convolutional Networks for Extensive Image Identification." The VGG16 model in ImageNet achieves approximately Top-5 test accuracy of 92.7%. With more than 14 million images, more than 1000 different types, ImageNet is a massive database. It was also one of the most often used models among those submitted at ILSVRC-2014. It performs significantly better when the huge kernel-sized filters are swapped out for many 3×3 kernel-sized filters one after the other.

3.2.4 MODULE 3: LeNET

The three main deep learning modules found in the small network LeNet are representedby the layers that stand for the convolutional, pooling, and complete link. It serves as a basis for additional deep learning models. This study presents a detailed analysis of LeNet5. You can concurrently have a better grasp of the pooling layer and the convolutional layer by employing example analysis. Let us examine the architecture of Lenet-5. Lenet-5 is the name of the network, which consists of five tiers with adjustable characteristics. All three of its convolution layer sets are combined using an average pooling combination. The mean pooling of and convolution layers are followed by two fully connected levels .A SoftMax classifier assigns a class to each image in the final step.

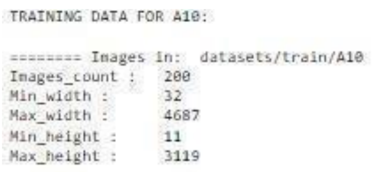
3.3 MODULE DESCRIPTION

3.3.1 IMPORT THE GIVEN IMAGE FROM DATASET:

We have to import our data set using keras preprocessing image data generator function also we create size, rescale, range, zoom range, horizontal flip. Then we import our image dataset from folder through the data generator function.Here we set train, test, and validation also we set target size, batch size and class- mode from this function we have to train using our own created network by addinglayers of CNN.

Military aircraft and Passenger aircraft:

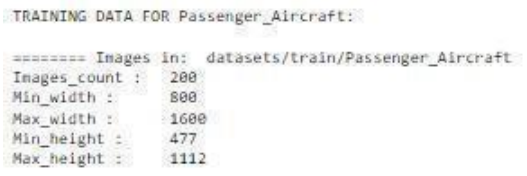
Military aircraft:



```
TRAINING DATA FOR A10:

======== Images in:  datasets/train/A10
Images_count :    200
Min_width :       32
Max_width :       4687
Min_height :      11
Max_height :      3119
```

Fig .3.1 SAMPLE DATASET MILITARY AIRCRAFT

Passenger aircraft:



```
TRAINING DATA FOR Passenger_Aircraft:

======== Images in:  datasets/train/Passenger_Aircraft
Images_count :   200
Min_width :      800
Max_width :      1600
Min_height :     477
Max_height :     1112
```

Fig .3.2 SAMPLE DATASET PASSENGER AIRCRAFT

3.3.2 TO TRAIN THE MODULE BY GIVEN IMAGE DATASET:

To train our dataset using a classifier and the `fit_generator` function, we first ensure that the dataset is properly formatted and split into training and validation sets. We then instantiate the classifier model, specifying its architecture and parameters. Utilizing data generators, we efficiently load batches of data for training, preprocessing it as necessary. Setting up training parameters involves defining the number of steps per epoch, the total number of epochs, validation data, and validation steps. With the model compiled using appropriate loss functions, optimizers, and evaluation metrics, we proceed to train it using the `fit_generator` function. Throughout training, the model iterates through the dataset, adjusting its parameters to minimize the loss function. Validation of the model's performance on the validation dataset helps assess its generalization ability and detect overfitting. Monitoring training progress by observing metrics such as loss and accuracy aids in fine-tuning hyperparameters if necessary. Finally, once training is complete, the trained model is saved for future use or deployment. This systematic approach ensures that our model learns to make accurate predictions effectively. In conclusion, training a dataset using a classifier and the `fit_generator` function involves several key steps to ensure the model learns effectively. By properly preparing the dataset, instantiating the classifier, setting up data generators, defining training parameters, and monitoring training progress, we can iteratively improve the model's performance. Validation of the model's performance on a separate dataset helps assess its generalization ability and avoid overfitting. Fine-tuning hyperparameters based on training progress further enhances the model's accuracy. Ultimately, saving the trained model allows for future deployment and use in various applications.

```
Model: "model"
______________________________________________________________________
Layer (type)                  Output Shape              Param #
==================================================================
input_1 (InputLayer)          [(None, 200, 200, 3)]     0
______________________________________________________________________
zero_padding2d (ZeroPadding2 (None, 206, 206, 3)        0
______________________________________________________________________
conv2d (Conv2D)               (None, 100, 100, 64)      9472
______________________________________________________________________
batch_normalization (BatchNo (None, 100, 100, 64)       256
______________________________________________________________________
activation (Activation)       (None, 100, 100, 64)      0
______________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 49, 49, 64)         0
______________________________________________________________________
conv2d_1 (Conv2D)             (None, 49, 49, 64)        4160
______________________________________________________________________
batch_normalization_1 (Batch (None, 49, 49, 64)         256
______________________________________________________________________
activation_1 (Activation)     (None, 49, 49, 64)        0
______________________________________________________________________
conv2d_2 (Conv2D)             (None, 49, 49, 64)        36928
______________________________________________________________________
batch_normalization_2 (Batch (None, 49, 49, 64)         256
______________________________________________________________________
activation_2 (Activation)     (None, 49, 49, 64)        0
______________________________________________________________________
conv2d_3 (Conv2D)             (None, 49, 49, 256)       16640
______________________________________________________________________
batch_normalization_3 (Batch (None, 49, 49, 256)        1024
______________________________________________________________________
activation_3 (Activation)     (None, 49, 49, 256)       0
______________________________________________________________________

______________________________________________________________________
batch_normalization_4 (Batch (None, 49, 49, 64)         256
______________________________________________________________________
activation_4 (Activation)     (None, 49, 49, 64)        0
______________________________________________________________________
conv2d_5 (Conv2D)             (None, 49, 49, 64)        36928
______________________________________________________________________
batch_normalization_5 (Batch (None, 49, 49, 64)         256
______________________________________________________________________
activation_5 (Activation)     (None, 49, 49, 64)        0
______________________________________________________________________
conv2d_6 (Conv2D)             (None, 49, 49, 256)       16640
______________________________________________________________________
batch_normalization_6 (Batch (None, 49, 49, 256)        1024
______________________________________________________________________
activation_6 (Activation)     (None, 49, 49, 256)       0
______________________________________________________________________
conv2d_7 (Conv2D)             (None, 49, 49, 64)        16448
______________________________________________________________________
batch_normalization_7 (Batch (None, 49, 49, 64)         256
______________________________________________________________________
```

TABLE 3.1 CNN Model Summary details

3.3.3 WORKING PROCESS OF LAYERS IN CNN MODEL:

A Convolutional Neural Network (ConvNet/CNN) is a groundbreaking Deep Learning algorithm designed to process images, extracting meaningful features and patterns for classification tasks. Its architecture is inspired by the connectivity patterns of neurons in the human brain, particularly the visual cortex. Like neurons in the brain responding to specific stimuli within their receptive fields, ConvNets employ filters to detect features in localized regions of an image. What sets ConvNets apart is their ability to automatically learn these filters through training, eliminating the need for hand-engineering. This feature makes ConvNets highly efficient in image processing tasks, requiring minimal preprocessing compared to traditional classification algorithms. The architecture typically comprises multiple layers, each refining the learned features to make accurate classifications. For instance, a basic ConvNet might consist of layers with varying numbers of units, such as 1,024 input units, 256 units in the first hidden layer, eight units in the second hidden layer, and finally, two output units for classification. Through extensive training, ConvNets demonstrate remarkable proficiency in image recognition, paving the way for advancements in various fields reliant on visual data analysis. In conclusion, Convolutional Neural Networks (ConvNets/CNNs) revolutionize image processing tasks by mimicking the connectivity patterns of neurons in the human visual cortex. With the ability to automatically learn features from data, ConvNets require minimal preprocessing compared to traditional methods.

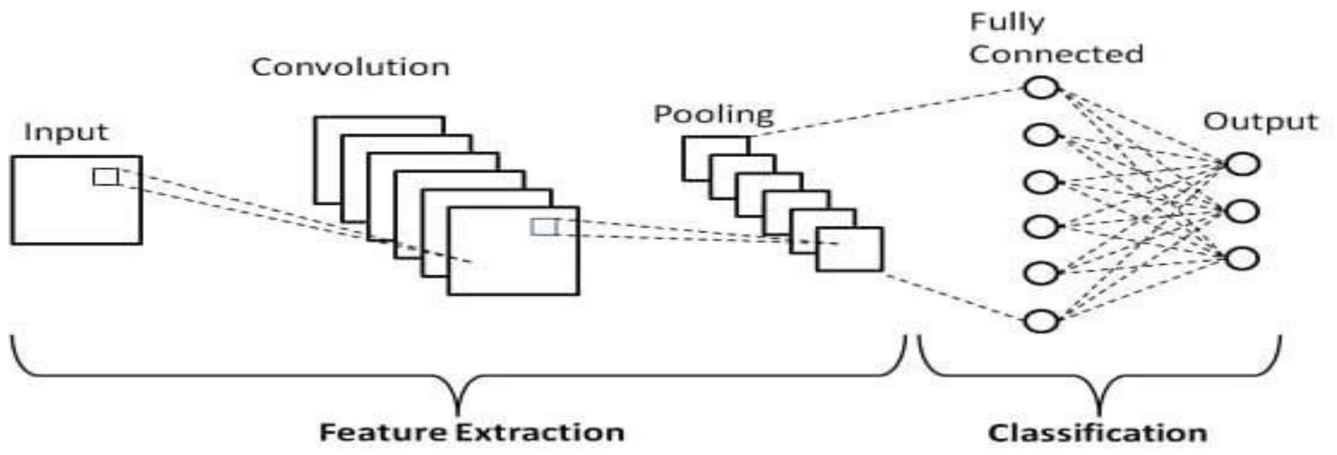


Fig .3.3 LAYERS IN CNN

Input Layer:

Input layer in CNN contain image data. Image data is represented by three dimensional matrixes. It needs to reshape it into a single column. Suppose you have image of dimension 28 x 28 =784, it need to convert it into 784 x 1 before feeding into input

Convo Layer:

The convolutional layer in a Convolutional Neural Network (ConvNet/CNN) is often referred to as the feature extractor layer due to its pivotal role in extracting meaningful features from input images. This layer operates by sliding learnable filters over local regions of the input image, known as receptive fields, and computing dot products between the filter weights and the corresponding values in these regions. This process generates activation values, representing detected features, which are then compiled into an output volume. By systematically scanning the entire image using a specified stride, the convolutional layer extracts relevant features across various spatial locations. The resulting output volume serves as input for subsequent layers, facilitating further feature abstraction and hierarchical representation learning. Through this iterative process, ConvNets can effectively discern complex patterns and achieve high accuracy in tasks such as image recognition and classification.

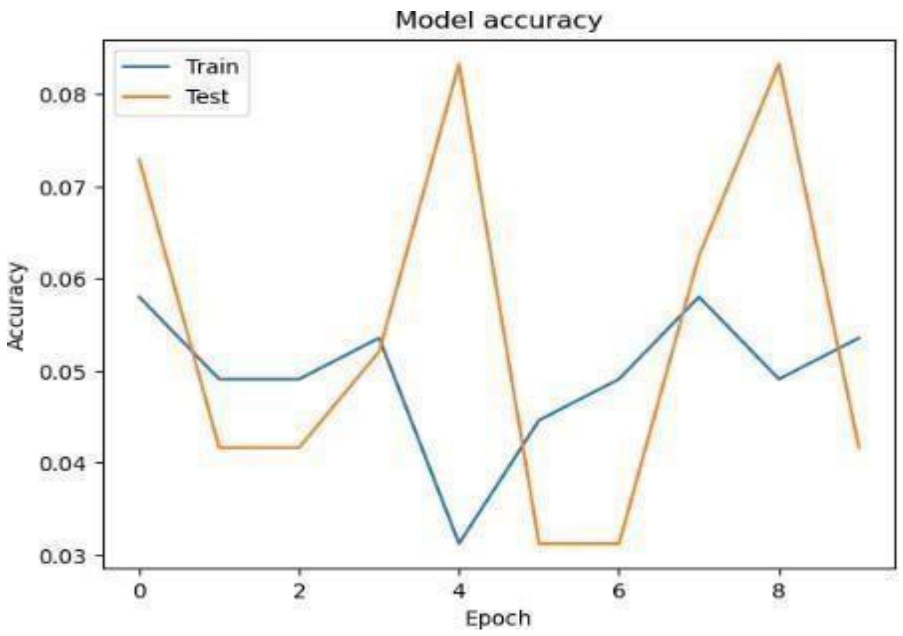


Fig.3.4 CNN model trained dataset accuracy

Pooling Layer:

Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layers. If it applies FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive. So, the max pooling is only way to reduce the spatial volume of input image. It has applied max pooling in single depth slice with Stride of 2. It can observe the 4 x 4 dimension input is reducingto 2 x 2 dimensions.

Fully Connected Layer (FC):

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different categories by training.

Softmax / Logistic Layer:

Softmax or Logistic layer is the last layer of CNN. It resides at the end of FC layer.

Logistic is used for binary classification and softmax is for multi-classification.

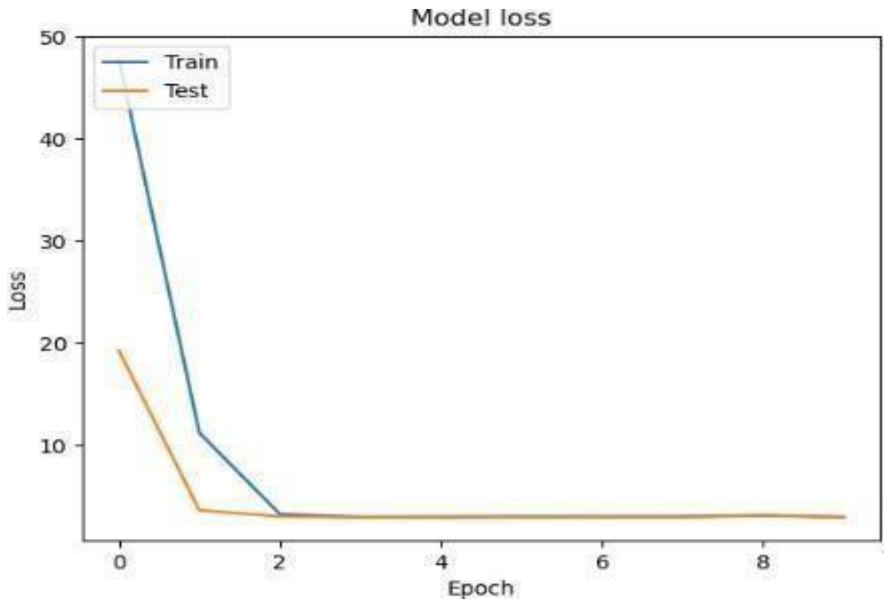


Fig.3.5 CNN model trained dataset loss values

Output Layer:

Output layer contains the label which is in the form of one-hot encoded. Now youhave a good understanding of CNN.

3.4 DATA FLOW DIAGRAM

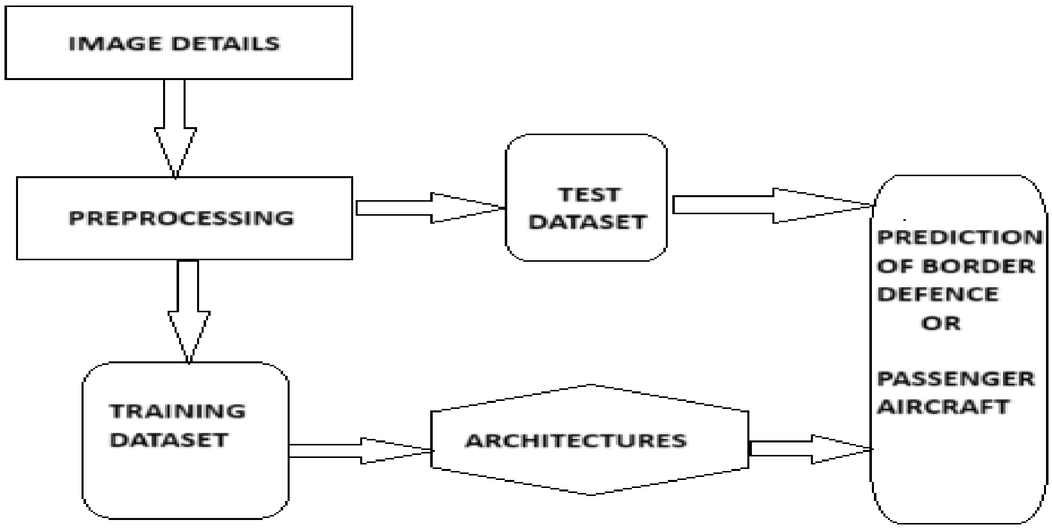


Fig .3.5 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) serves as a comprehensive visual representation of the intricate pathways through which data traverses within an information system, offering a holistic view of processes, data sources, destinations, and storage entities. Unlike flowcharts, DFDs focus exclusively on illustrating data processing, omitting control flow or process sequence details. Utilizing various symbols such as those for external entities, processes, data stores, and data flows, DFDs provide a structured framework for understanding system functionality and serveas a foundational element in system design. DFDs can be organized into multiple levels, with each level offering progressively detailed insights into system operations. From a high-level overview at Level 0 to more granular depictions at subsequent levels, the depth of detail within DFDs is tailored to the specific scope of analysis. Crucially, DFDs foster effective communication among stakeholders and play an indispensable role in system analysis, design, and communication endeavors, facilitating a shared understanding of complex information systems. In essence, Data Flow Diagrams (DFDs) provide a structured and comprehensive visualization of data pathways within an information system, focusing solely on data processing without delving into control flow or process sequencing. By employing various symbols to represent entities, processes, and data flows, DFDs offer a clear framework for understanding system functionality.

3.5 SYSTEM ARCHITECTURE

In the development of a Convolutional Neural Network (CNN) system for predicting border defense mechanisms, a methodical process is followed, beginning with thorough data collection, analysis, and preprocessing to ensure dataset suitability. Subsequently, various CNN architectures undergo rigorous comparison and evaluation to identify the most optimal model, which is then fine-tuned for accuracy through hyperparameter optimization. Once the finalized model architecture is determined, extensive training using preprocessed data is conducted. Upon successful training, the CNN model is seamlessly integrated into a Django web application framework for efficient deployment, enabling real-time predictions of border defense mechanisms. This comprehensive approach not only enhances security measures but also equips stakeholders with actionable insights for informed decision-making in safeguarding national borders. The development of a Convolutional Neural Network (CNN) system for border defense involves meticulous data collection, model selection, training, and integration into a web application framework. Through this approach, stakeholders are equipped with real-time predictive capabilities, enhancing security measures and informed decision-making in safeguarding national borders.

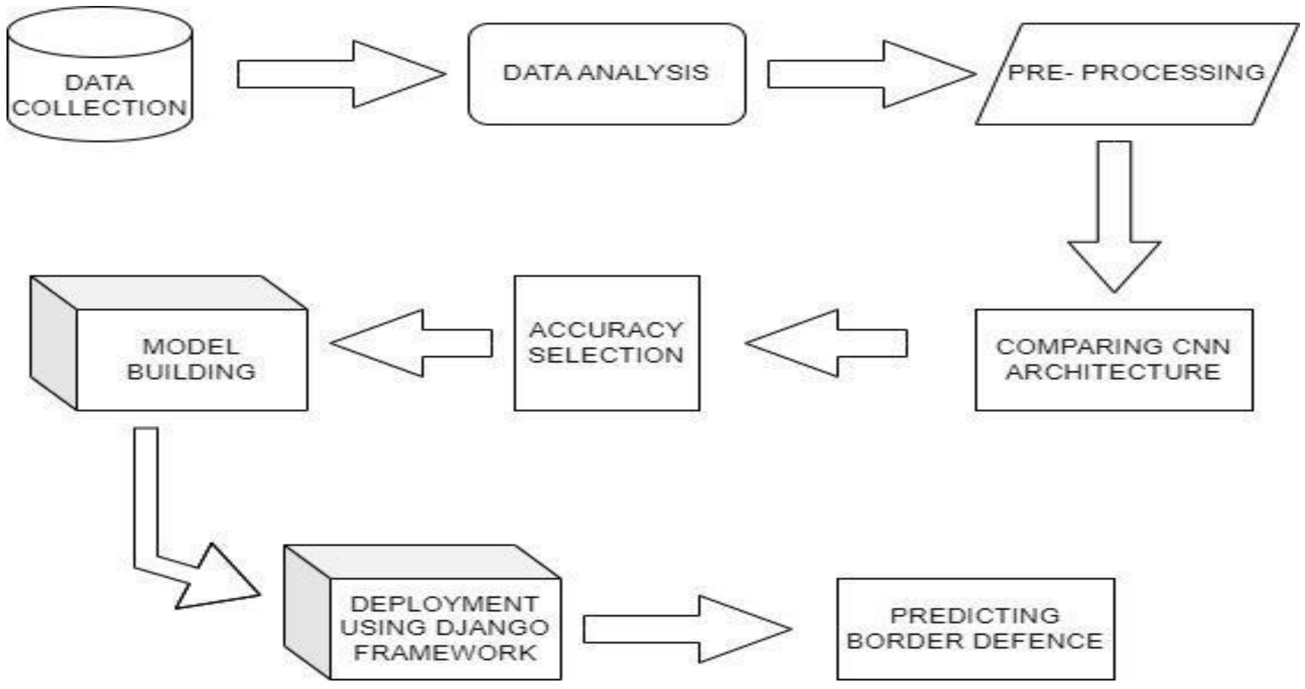

DATA COLLECTION
DATA ANALYSIS
PRE- PROCESSING
COMPARING CNN ARCHITECTURE
ACCURACY SELECTION
MODEL BUILDING
DEPLOYMENT USING DJANGO FRAMEWORK
PREDICTING BORDER DEFENCE

Fig .3.7 SYSTEM ARCHITECTURE

3.6 WORK FLOW DIAGRAM

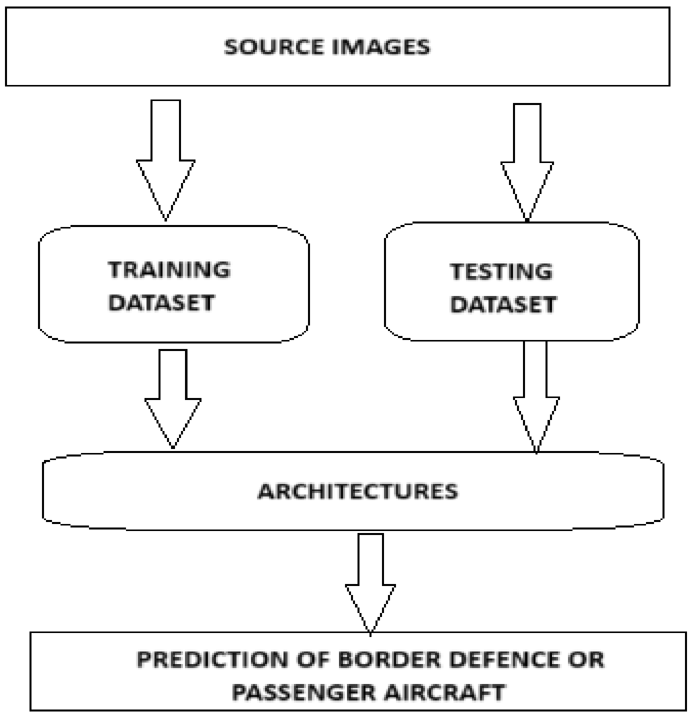


Fig .3.8 WORK FLOW DIAGRAM

The workflow for developing a Convolutional Neural Network (CNN) system for predictingborder defense mechanisms or passenger aircraft begins with sourcing a diverse dataset containing images representative of the target classes. This dataset is then split into two subsets: a training dataset and a testing dataset, facilitating model training and evaluation. Subsequently, both datasets undergo preprocessing to enhance their quality and suitability for training, involving tasks such as resizing, normalization, and augmentation. Following this, various CNN architectures are explored and compared to select the most appropriate model for the task. The chosen architecture is then trained using the preprocessed training dataset, enabling the model to learn to extract relevant features and classify images accurately. Upon completion of training, the model's performance is evaluated using the testing dataset, assessing metrics such as accuracy and precision. Once deemed satisfactory,the trained CNN model is deployed for real-time prediction, potentially integrating it into a web application or deployment platform. Finally, the deployed model can be utilized to makepredictions on new images, effectively identifying and classifying border defense mechanisms or passenger aircraft with accuracy and efficiency

3.7 UML DIAGRAM

3.7.1 USE CASE DIAGRAM

Use case diagrams are essential visual tools in system analysis, illustrating interactions between actors and use cases to represent system functionalities. By encapsulating user-system interactions, they provide a clear understanding of how users interact with the system. These diagrams aid in requirement gathering, system design, and communication among stakeholders. They facilitate the identification of system boundaries and help prioritize features based on user needs. Use case diagrams also support iterative development, allowing for continuous refinement and improvement of the system's design.

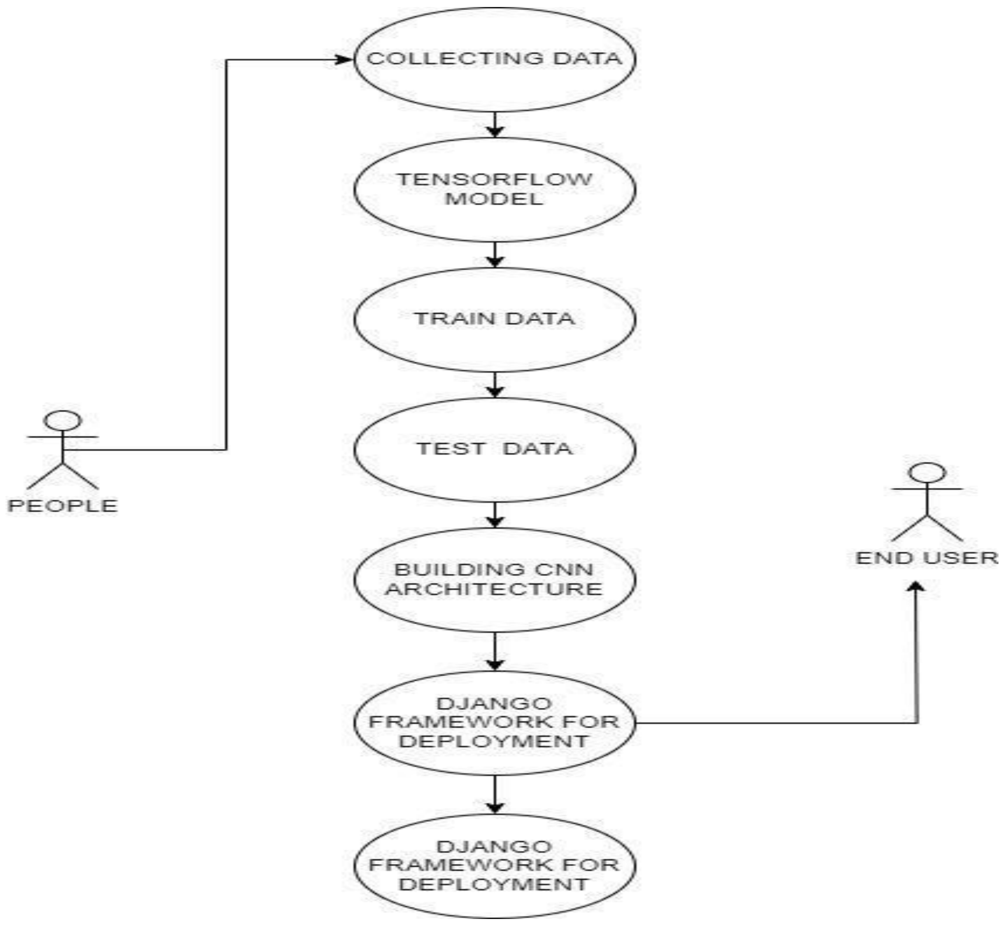

PEOPLE
COLLECTING DATA
TENSORFLOW MODEL
TRAIN DATA
TEST DATA
BUILDING CNN ARCHITECTURE
DJANGO FRAMEWORK FOR DEPLOYMENT
DJANGO FRAMEWORK FOR DEPLOYMENT
END USER

Fig .3.9 USECASE DIAGRAM

3.7.2 CLASS DIAGRAM

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

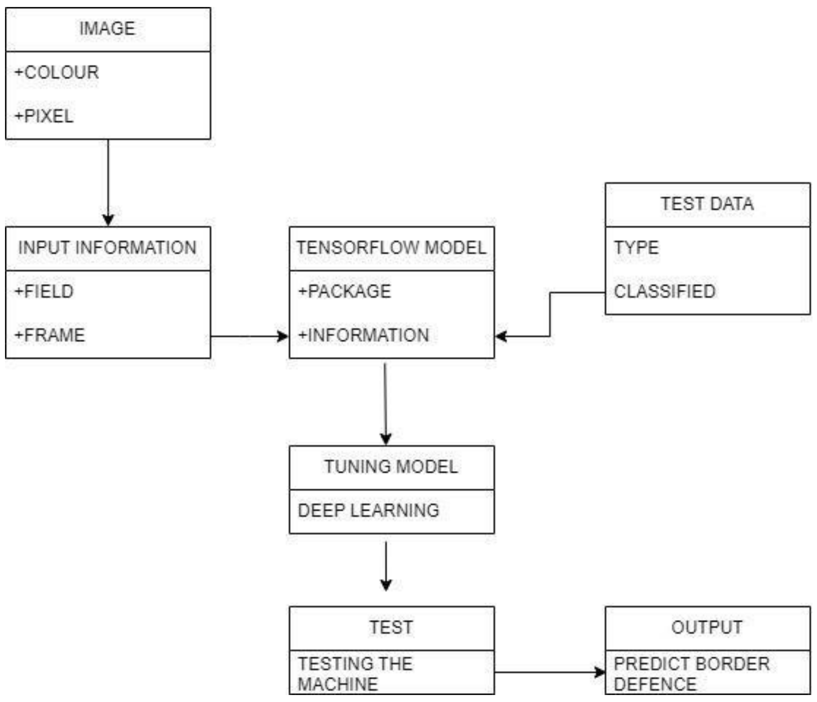


Fig .3.10 CLASS DIAGRAM

3.7.3 ACTIVITY DIAGRAM

The activity diagram in UML is used to show the system's control flow rather than its implementation. Both concurrent and sequential activities are modelled. The workflow from one action to the next can be seen using the activity diagram. It p laced emphasis on the existence of flow and the sequence in which it takes place. The flow may be sequential, branching, or concurrent, and the activity diagram has been designed with a fork, join, etc. to deal with these many types of flows. A flowchart thatis object- oriented is another name for it. It includes tasks that include applying a seriesof actionsor processes to simulate the behavioral diagram. Activity diagrams in UML visualize the control flow of a system, focusing on the sequence of actions rather than their implementation details. They accommodate sequential, branching, and concurrent flows, employing constructs like fork and join to represent these variations.

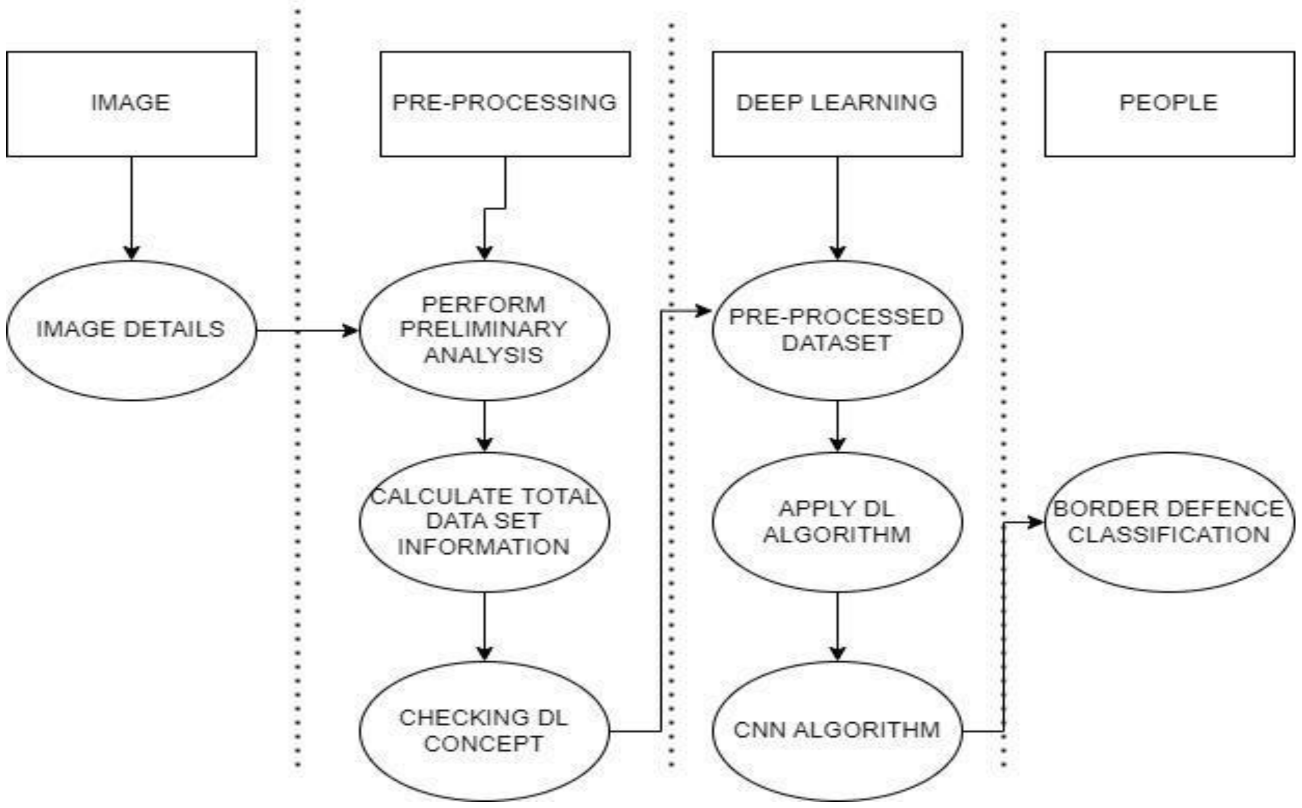


Fig .3.11 ACTIVITY DIAGRAM

3.7.4 SEQUENCE DIAGRAM

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching. Sequence diagrams in UML offer a visual depiction of message flow within a system, also known as event diagrams. They facilitate the exploration of dynamic scenarios, showcasing communication between lifelines in a time-ordered sequence during runtime. Lifelines are depicted as vertical bars, while message flow is represented by vertical dotted lines at the bottom of the diagram.

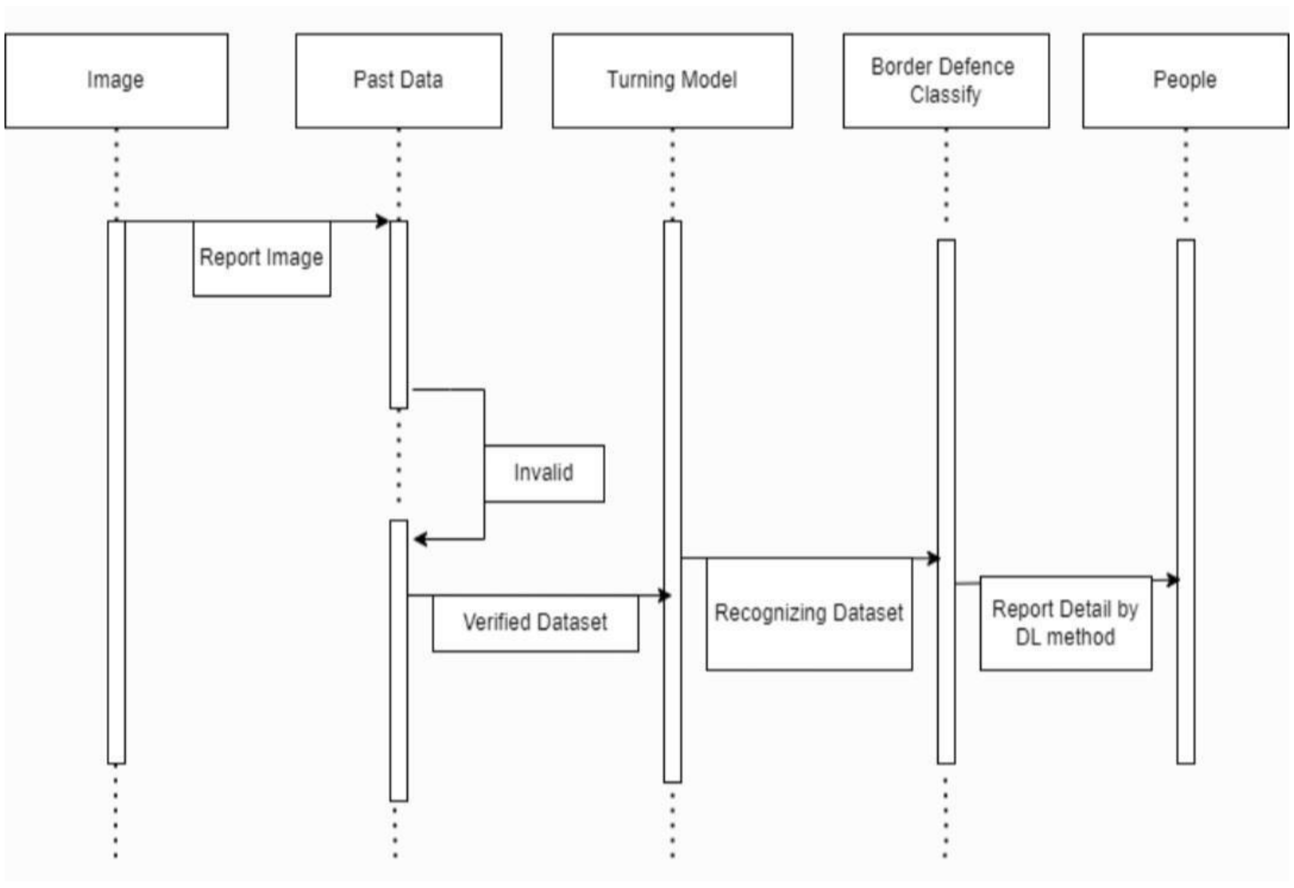


Fig .3.12 SEQUENCE DIAGRAM

CHAPTER – 4
REQUIREMENT SPECIFICATION

4.1 HARDWARE REQUIREMENTS:

- Processor : Intel i3
- Hard disk : minimum 400 GB
- RAM : minimum 4 GB

4.2 SOFTWARE REQUIREMENTS:

- Python: High-level, general-purpose programming language emphasizing readability and versatility. Supports multiple paradigms and boasts a comprehensive standard library.
- Anaconda: Distribution of Python and R for scientific computing, simplifying package management and deployment. Developed by Anaconda, Inc., with editions for individuals, teams, and enterprises.
- Jupyter Notebook: Open-source web app enabling creation and sharing of interactive documents blending code, visualizations, and text. Facilitates collaboration and reproducible research.
- TensorFlow: Free and open-source library for machine learning and AI, developed by Google Brain Team. Focuses on training and deploying deep neural networks across various platforms.
- Keras: Open-source neural network library in Python, providing a user-friendly interface for building and experimenting with artificial neural networks. Originally supported multiple backends, now focused on TensorFlow.

In conclusion, the Python ecosystem, encompassing Anaconda, Jupyter Notebook, TensorFlow, and Keras, offers a versatile and efficient platform for machine learning and AI development. With Python's readability and Anaconda's simplified package management, Jupyter Notebook facilitates collaborative and reproducible research. TensorFlow's power in training and deploying deep neural networks, along with Keras' user-friendly interface, streamlines the development process

CHAPTER – 5
IMPLEMENTATION

5.1 SAMPLE CODE:

CODING:

Module 1:

```
import os
import numpy as np # linear algebra
import matplotlib.pyplot as plt

# Dl framwork - tensorflow, keras a backend
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Input, Dense, Flatten, Dropout, BatchNormalization
from tensorflow.keras.layers import Conv2D, SeparableConv2D, MaxPool2D, LeakyReLU, Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
from IPython.display import display
from os import listdir
from os.path import isfile, join
from PIL import Image
import glob
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense

import warnings
warnings.filterwarnings('ignore')

A10 = 'datasets/train/A10'
B52 = 'datasets/train/B52'
C17 = 'datasets/train/C17'
F4 = 'datasets/train/F4'
Passenger_Aircraft = 'datasets/train/Passenger_Aircraft'

def plot_images(item_dir, n=6):
    all_item_dir = os.listdir(item_dir)
    item_files = [os.path.join(item_dir, file) for file in all_item_dir][:n]

    plt.figure(figsize=(80, 40))
    for idx, img_path in enumerate(item_files):
        plt.subplot(7, n, idx+1)
        img = plt.imread(img_path)
        plt.imshow(img, cmap='gray')
        plt.axis('off')

    plt.tight_layout()


def Images_details_Print_data(data, path):
    print(" ====== Images in: ", path)
    for k, v in data.items():
        print("%s:\t%s" % (k, v))

def Images_details(path):
    files = [f for f in glob.glob(path + "**/*.*", recursive=True)]
    data = {}
    data['images_count'] = len(files)
    data['min_width'] = 10**100  # No image will be bigger than that
    data['max_width'] = 0
    data['min_height'] = 10**100  # No image will be bigger than that
    data['max_height'] = 0



    for f in files:
        im = Image.open(f)
        width, height = im.size
        data['min_width'] = min(width, data['min_width'])
        data['max_width'] = max(width, data['max_height'])
        data['min_height'] = min(height, data['min_height'])
        data['max_height'] = max(height, data['max_height'])

    Images_details_Print_data(data, path)

print("")
print("TRAINING DATA FOR A10:")
print("")
images_details(A10)
print("")
plot_images(A10, 10)

print("")
print("TRAINING DATA FOR B52:")
print("")
images_details(B52)
print("")
plot_images(B52, 10)

print("")
print("TRAINING DATA FOR C17:")
print("")
images_details(C17)
print("")
```

```
plot_images(C17, 10)

print("")
print("TRAINING DATA FOR F4:")
print("")
images_details(F4)
print("")
plot_images(F4, 10)

print("")
print("TRAINING DATA FOR Passenger_Aircraft:")
print("")
images_details(Passenger_Aircraft)
print("")
plot_images(Passenger_Aircraft, 10)


Classifier=Sequential()
Classifier.add(Convolution2D(32,(3,3),input_shape=(250,250,3),activation='relu'))
Classifier.add(MaxPooling2D(pool_size=(2,2)))
Classifier.add(Flatten())
Classifier.add(Dense(38, activation='relu'))


Classifier.add(Dense(2, activation='softmax'))
Classifier.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])


train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)


training_set=train_datagen.flow_from_directory('dataset/Train',target_size=(250,250),batch_size=32,class_m
ode='categorical')
test_set=test_datagen.flow_from_directory('dataset/Test',target_size=(250,250),batch_size=32,class_mode='c
ategorical')


epochs = 10
batch_size = 512


#### Fitting the model
history = Classifier.fit(
        training_set, steps_per_epoch=training_set.samples // batch_size,
        epochs=epochs,
        validation_data=test_set,validation_steps=test_set.samples // batch_size)


def plot():
    #Plot training & validation accuracy values
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Model accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()

    # Plot training & validation loss values
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()
plot()
```

Module 2:

VGG:

```
import warnings
warnings.filterwarnings('ignore')


import tensorflow
import tensorflow as tf
print(tf.__version__)

import keras
import keras.backend as K
from keras.models import Model
from keras.layers import Input, Dense, Conv2D, Conv3D, DepthwiseConv2D, SeparableConv2D,
Conv3DTranspose
from keras.layers import Flatten, MaxPool2D, AvgPool2D, GlobalAvgPool2D, UpSampling2D,
BatchNormalization
from keras.layers import Concatenate, Add, Dropout, ReLU, Lambda, Activation, LeakyReLU, PReLU

from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot

from time import time
import numpy as np

from keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import EarlyStopping

import warnings
warnings.filterwarnings('ignore')
```

```
2.6.0
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True,validation
_split = 0.2)
train_data=train.flow_from_directory(directory = 'datasets/train',target_size=(224,224),
                                    batch_size=32,class_mode='categorical')
Found 4000 images belonging to 20 classes.

test=ImageDataGenerator(rescale=1./255)
test_data=test.flow_from_directory(directory = 'datasets/test',target_size=(224,224),
                                  batch_size=32,class_mode='categorical')
Found 2000 images belonging to 20 classes.

def vgg(input_shape, n_classes):

  input = Input(input_shape)

  x = Conv2D(64, 3, padding='same', activation='relu')(input)
  x = Conv2D(64, 3, padding='same', activation='relu')(x)
  x = MaxPool2D(2, strides=2, padding='same')(x)

  x = Conv2D(128, 3, padding='same', activation='relu')(x)
  x = Conv2D(128, 3, padding='same', activation='relu')(x)
  x = MaxPool2D(2, strides=2, padding='same')(x)

  x = Conv2D(256, 3, padding='same', activation='relu')(x)
  x = Conv2D(256, 3, padding='same', activation='relu')(x)
  x = Conv2D(256, 3, padding='same', activation='relu')(x)
  x = MaxPool2D(2, strides=2, padding='same')(x)

  x = Conv2D(512, 3, padding='same', activation='relu')(x)
  x = Conv2D(512, 3, padding='same', activation='relu')(x)
  x = Conv2D(512, 3, padding='same', activation='relu')(x)
  x = MaxPool2D(2, strides=2, padding='same')(x)

  x = Conv2D(512, 3, padding='same', activation='relu')(x)
  x = Conv2D(512, 3, padding='same', activation='relu')(x)
  x = Conv2D(512, 3, padding='same', activation='relu')(x)
  x = MaxPool2D(2, strides=2, padding='same')(x)

  x = Flatten()(x)
  x = Dense(4096, activation='relu')(x)
  x = Dense(4096, activation='relu')(x)

  output = Dense(n_classes, activation='softmax')(x)

  model = Model(input, output)

model.compile(optimizer='Adam',loss='categorical_crossentropy',metrics=['accuracy',tensorflow.keras.metric
s.Precision()])
  return model

input_shape = 224, 224, 3
```

```
n_classes = 20

K.clear_session()
model = vgg(input_shape, n_classes)
model.summary()

model_path = "ALEXNET.h5"

from keras.callbacks import ModelCheckpoint

M = ModelCheckpoint(model_path, monitor='accuracy', verbose=1, save_best_only=True)

epochs = 100
batch_size = 512

#### Fitting the model
history = model.fit(
        train_data, steps_per_epoch=train_data.samples // batch_size,
        epochs=epochs,
        validation_data=test_data,validation_steps=test_data.samples // batch_size,
        callbacks=[M])
history.history.keys()

dict_keys(['loss', 'accuracy', 'precision', 'val_loss', 'val_accuracy', 'val_precision'])

import matplotlib.pyplot as plt
import numpy as np

plt.figure(figsize=(20, 8))
plt.plot(history.history['accuracy'])

for i in range(epochs):
    if i%5 == 0:
        plt.annotate(np.round(history.history['accuracy'][i]*100,2),xy=(i,history.history['accuracy'][i]))

plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.show()
plt.figure(figsize=(20, 8))
plt.plot(history.history['loss'])

for i in range(epochs):
    if i%5 == 0:
        plt.annotate(np.round(history.history['loss'][i]*100,2),xy=(i,history.history['loss'][i]))

plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.show()
```

Module 3:

LENET:

```
import warnings
warnings.filterwarnings('ignore')


# import os
# os.environ['KMP_DUPLICATE_LIB_OK']="TRUE"

import tensorflow
import tensorflow as tf
print(tf.__version__)

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPool2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Activation
```

```
2.6.0
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True,validation
_split = 0.2)
train_data=train.flow_from_directory(directory = 'datasets/train',target_size=(224,224),
                                   batch_size=32,class_mode='categorical')
```

```
Found 4000 images belonging to 20 classes.
```

```
test=ImageDataGenerator(rescale=1./255)
test_data=test.flow_from_directory(directory = 'datasets/test',target_size=(224,224),
                                 batch_size=32,class_mode='categorical')
```

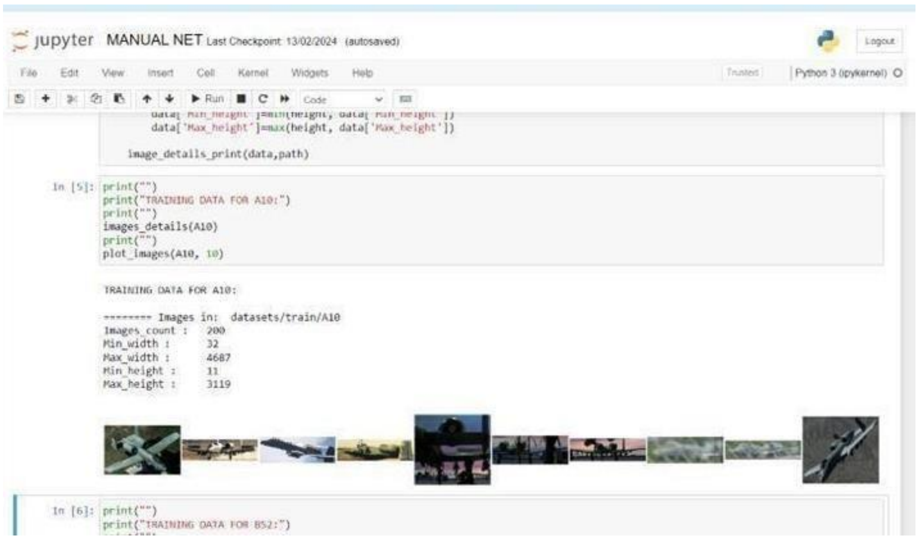
```
Found 2000 images belonging to 20 classes.
```

```
MODEL=Sequential()
MODEL.add(Convolution2D(filters=32, kernel_size=(3,3), strides=(3,3), input_shape=(224,224,3),
padding=('valid'), activation='relu'))
MODEL.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
MODEL.add(Convolution2D(filters=128, kernel_size=(3,3), strides=(3,3), padding=('valid'),
MODEL.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
MODEL.add(Flatten())
MODEL.add(Dense(256, activation='relu'))
MODEL.add(Dense(20, activation='softmax'))

OPT    = tensorflow.keras.optimizers.Adam(0.001)

MODEL.compile(optimizer=OPT,loss='categorical_crossentropy',metrics=["accuracy",
tensorflow.keras.metrics.Precision(), tensorflow.keras.metrics.Recall()])
MODEL.summary()
model_path = "LENET.h5"

from tensorflow.keras.callbacks import ModelCheckpoint

M = ModelCheckpoint(model_path, monitor='accuracy', verbose=1, save_best_only=True, mode='max')


epochs = 100
batch_size = 32


WORKING = MODEL.fit_generator(
        train_data, steps_per_epoch=train_data.samples // batch_size,
        epochs=epochs,
        validation_data=test_data,validation_steps=test_data.samples // batch_size,
        callbacks=[M])
import matplotlib.pyplot as plt
import numpy as np

plt.figure(figsize=(20, 8))
plt.plot(WORKING.history['accuracy'])

for i in range(epochs):
   if i%5 == 0:

plt.annotate(np.round(WORKING.history['accuracy'][i]*100,2),xy=(i,WORKING.history['accuracy'][i]))

plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.show()
import matplotlib.pyplot as plt

plt.figure(figsize=(20, 8))
plt.plot(WORKING.history['loss'])

plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.show()
```

CHAPTER-6

RESULT AND ANALYSIS

MANUAL DATASET:

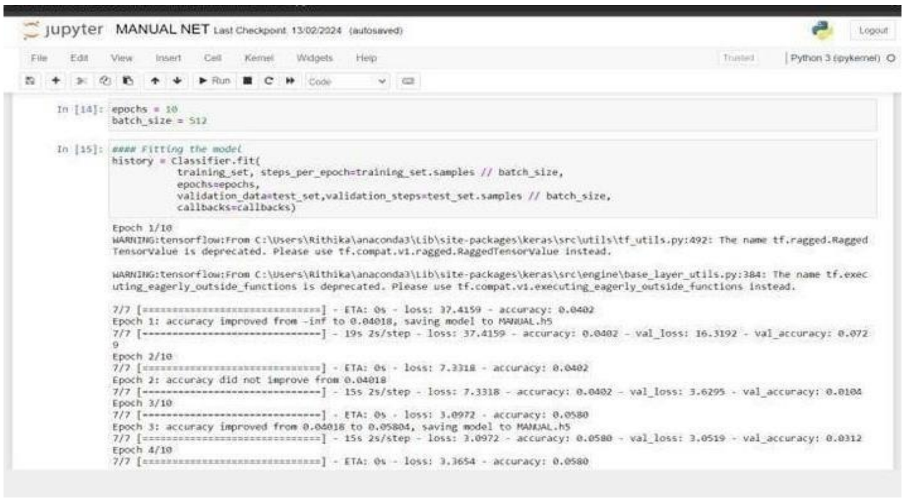

jupyter MANUAL NET Last Checkpoint: 13/02/2024 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
        data['Max_height']=max(height, data['Max_height'])

    image_details_print(data,path)

In [5]: print("")
        print("TRAINING DATA FOR A10:")
        print("")
        images_details(A10)
        print("")
        plot_images(A10, 10)

        TRAINING DATA FOR A10:

        ======== Images in:  datasets/train/A10
        Images_count :   200
        Min_width :      32
        Max_width :      4687
        Min_height :     11
        Max_height :     3119

In [6]: print("")
        print("TRAINING DATA FOR B52:")
```

Fig 6.1 MANUAL DATASET

The manual dataset is likely a subset of the overall dataset, possibly containing images or data points that require human verification, correction, or annotation. This dataset would serve as a crucial component in refining and augmenting the system's performance, as mentioned in the manual module description.

MANUAL ACCURACY RATE:

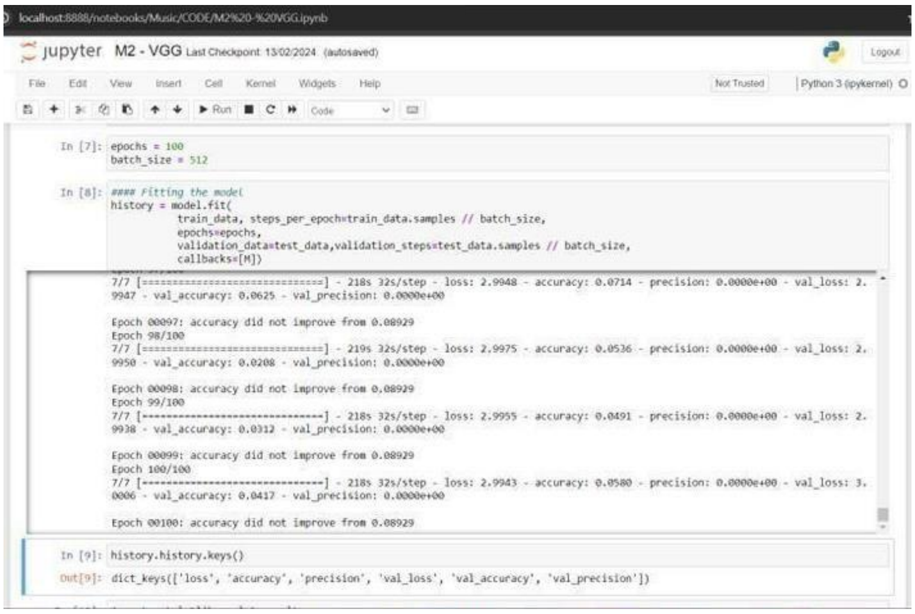

jupyter MANUAL NET Last Checkpoint: 13/02/2024 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [14]: epochs = 10
         batch_size = 512

In [15]: #### Fitting the model
         history = Classifier.fit(
                     training_set, steps_per_epoch=training_set.samples // batch_size,
                     epochs=epochs,
                     validation_data=test_set,validation_steps=test_set.samples // batch_size,
                     callbacks=callbacks)

         Epoch 1/10
         WARNING:tensorflow:From C:\Users\Rithika\anaconda3\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.Ragged
         TensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

         WARNING:tensorflow:From C:\Users\Rithika\anaconda3\Lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.exec
         uting_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

         7/7 [==============================] - ETA: 0s - loss: 37.4159 - accuracy: 0.0402
         Epoch 1: accuracy improved from -inf to 0.04018, saving model to MANUAL.h5
         7/7 [==============================] - 19s 2s/step - loss: 37.4159 - accuracy: 0.0402 - val_loss: 16.3192 - val_accuracy: 0.072
         9
         Epoch 2/10
         7/7 [==============================] - ETA: 0s - loss: 7.3318 - accuracy: 0.0402
         Epoch 2: accuracy did not improve from 0.04018
         7/7 [==============================] - 15s 2s/step - loss: 7.3318 - accuracy: 0.0402 - val_loss: 3.6295 - val_accuracy: 0.0104
         Epoch 3/10
         7/7 [==============================] - ETA: 0s - loss: 3.0972 - accuracy: 0.0580
         Epoch 3: accuracy improved from 0.04018 to 0.05804, saving model to MANUAL.h5
         7/7 [==============================] - 15s 2s/step - loss: 3.0972 - accuracy: 0.0580 - val_loss: 3.0519 - val_accuracy: 0.0312
         Epoch 4/10
         7/7 [==============================] - ETA: 0s - loss: 3.3654 - accuracy: 0.0580
```

Fig 6. 2 MANUAL ACCURACY RATE

The accuracy rate of the manual dataset would depend on the effectiveness of human verification, correction, and annotation processes. Since human judgment and expertise are integrated into the system, the accuracy rate could be quite high, assuming that the manual tasks are performed diligently and accurately.

VGG ACCURACY RATE:

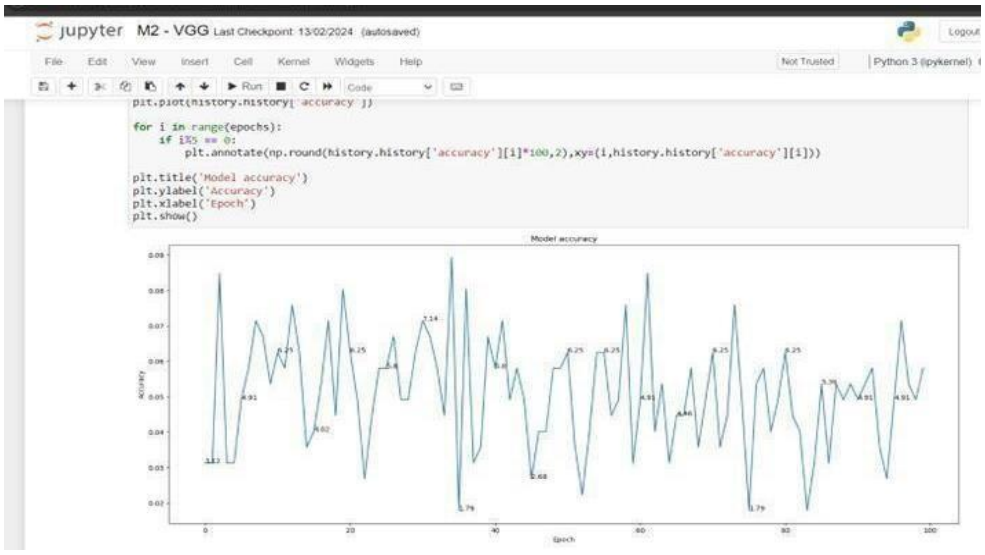


```
localhost:8888/notebooks/Music/CODE/M2%20-%20VGG.ipynb
jupyter M2 - VGG Last Checkpoint: 13/02/2024 (autosaved)

In [7]: epochs = 100
        batch_size = 512

In [8]: #### Fitting the model
        history = model.fit(
                    train_data, steps_per_epoch=train_data.samples // batch_size,
                    epochs=epochs,
                    validation_data=test_data,validation_steps=test_data.samples // batch_size,
                    callbacks=[M])

        7/7 [==============================] - 218s 32s/step - loss: 2.9948 - accuracy: 0.0714 - precision: 0.0000e+00 - val_loss: 2.
        9947 - val_accuracy: 0.0625 - val_precision: 0.0000e+00

        Epoch 00097: accuracy did not improve from 0.08929
        Epoch 98/100
        7/7 [==============================] - 219s 32s/step - loss: 2.9975 - accuracy: 0.0536 - precision: 0.0000e+00 - val_loss: 2.
        9950 - val_accuracy: 0.0208 - val_precision: 0.0000e+00

        Epoch 00098: accuracy did not improve from 0.08929
        Epoch 99/100
        7/7 [==============================] - 218s 32s/step - loss: 2.9955 - accuracy: 0.0491 - precision: 0.0000e+00 - val_loss: 2.
        9938 - val_accuracy: 0.0312 - val_precision: 0.0000e+00

        Epoch 00099: accuracy did not improve from 0.08929
        Epoch 100/100
        7/7 [==============================] - 218s 32s/step - loss: 2.9943 - accuracy: 0.0580 - precision: 0.0000e+00 - val_loss: 3.
        0006 - val_accuracy: 0.0417 - val_precision: 0.0000e+00

        Epoch 00100: accuracy did not improve from 0.08929

In [9]: history.history.keys()
Out[9]: dict_keys(['loss', 'accuracy', 'precision', 'val_loss', 'val_accuracy', 'val_precision'])
```

Fig 6.3 VGG ACCUARCY RATE

The VGG architecture is widely recognized for its prowess in image classification tasks due to its deep convolutional neural network design. With its intricate feature extraction capabilities, VGG is well-suited to accurately classify images across predefined categories. By comparing the model's predictions with the ground truth labels in the test dataset, we can calculate its accuracy rate. This evaluation process provides valuable insights into the model's performance and its ability to generalize to unseen data. With its proven track record in achieving high accuracy rates, the VGG architecture remains a reliable choice for various computer vision applications.

VGG OVERALL ACCURACY GRAPH:

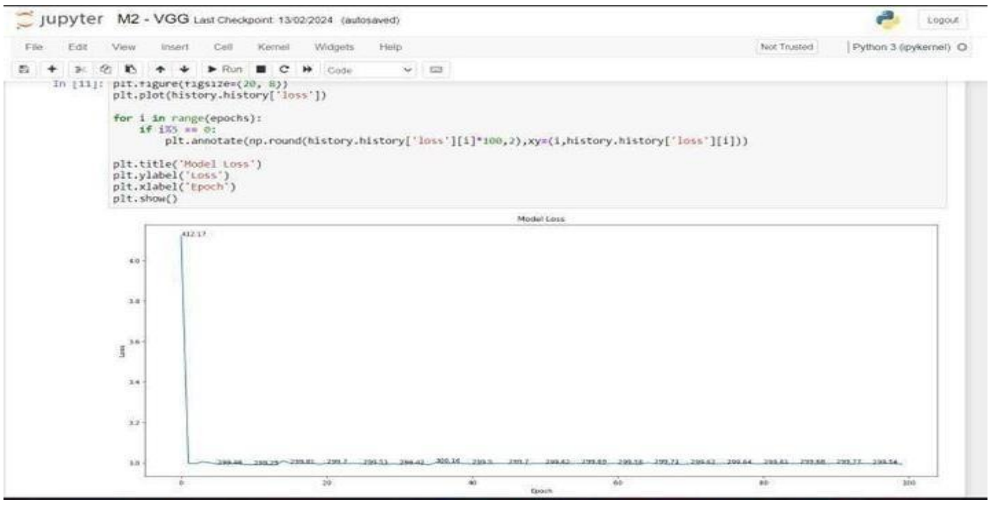

jupyter M2 - VGG Last Checkpoint: 13/02/2024 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Logout
Run
Code
Model accuracy
Accuracy
Epoch

```
plt.plot(history.history['accuracy'])

for i in range(epochs):
    if i%5 == 0:
        plt.annotate(np.round(history.history['accuracy'][i]*100,2),xy=(i,history.history['accuracy'][i]))

plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.show()
```

Fig 6.4 VGG OVER ALL ACCUARY RATE

The overall accuracy graph illustrates the performance of the border defense mechanism classification system over multiple training epochs. Each epoch represents a complete pass through the entire training dataset during the model training process.The accuracy graph typically shows an increasing trend during the initial epochs as the model learns from the training data. However, it may plateau or fluctuate in later epochs as the model converges or encounters overfitting.

VGG OVERALL LOSS GRAPH:


jupyter M2 - VGG Last Checkpoint: 13/02/2024 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Logout
Run
Code
Model Loss
412.17
Loss
Epoch

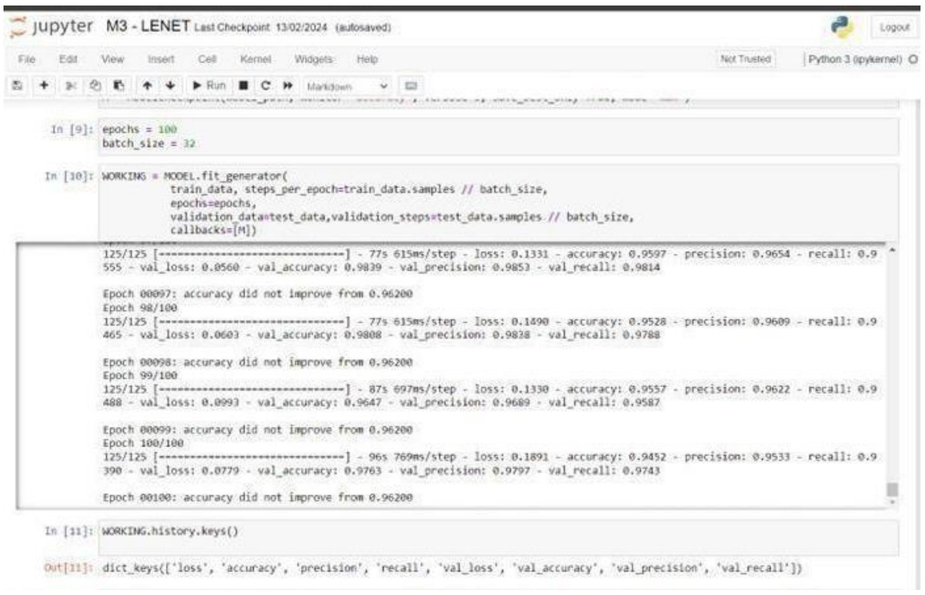
```
In [11]: plt.figure(figsize=(20, 8))
         plt.plot(history.history['loss'])

         for i in range(epochs):
             if i%5 == 0:
                 plt.annotate(np.round(history.history['loss'][i]*100,2),xy=(i,history.history['loss'][i]))

         plt.title('Model Loss')
         plt.ylabel('Loss')
         plt.xlabel('Epoch')
         plt.show()
```

Fig 6.5 VGG OVERALL LOSS GRAPH

The VGG overall loss graph represents the model's loss function over multiple training epochs. The loss function measures the difference between the predicted outputs of the model and the actual ground truth labels in the training data.

The loss graph typically shows a decreasing trend during the training process, indicating that the model is gradually minimizing its error. However, it may exhibit fluctuations or spikes due to variations in the training data or model architecture.

LENET ACCURACY RATE:



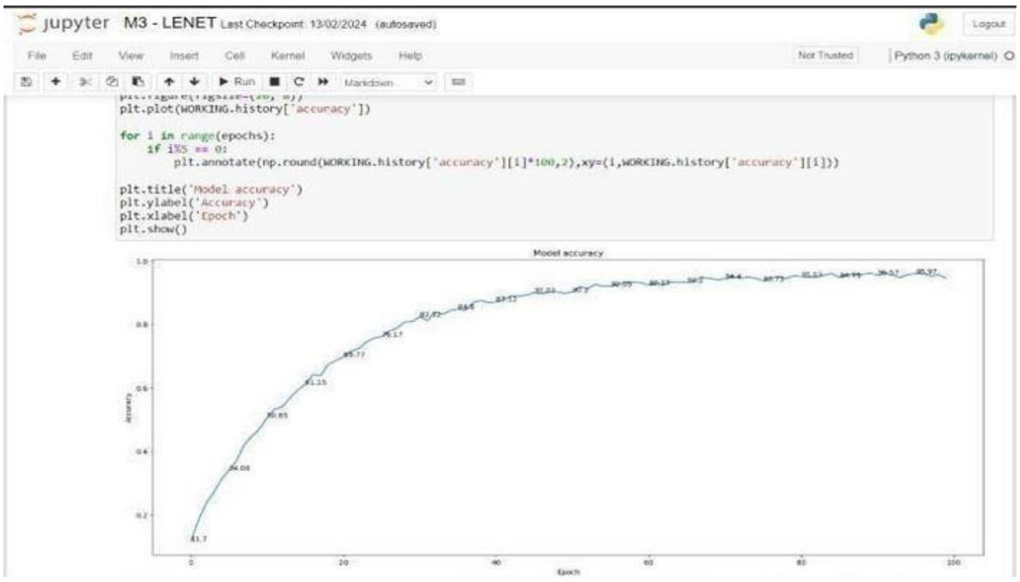
jupyter M3 - LENET Last Checkpoint 13/02/2024 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

```
In [9]: epochs = 100
        batch_size = 32

In [10]: WORKING = MODEL.fit_generator(
                     train_data, steps_per_epoch=train_data.samples // batch_size,
                     epochs=epochs,
                     validation_data=test_data,validation_steps=test_data.samples // batch_size,
                     callbacks=[M])

125/125 [==============================] - 77s 615ms/step - loss: 0.1331 - accuracy: 0.9597 - precision: 0.9654 - recall: 0.9
555 - val_loss: 0.0560 - val_accuracy: 0.9839 - val_precision: 0.9853 - val_recall: 0.9814

Epoch 00097: accuracy did not improve from 0.96200
Epoch 98/100
125/125 [==============================] - 77s 615ms/step - loss: 0.1490 - accuracy: 0.9528 - precision: 0.9609 - recall: 0.9
465 - val_loss: 0.0603 - val_accuracy: 0.9808 - val_precision: 0.9838 - val_recall: 0.9788

Epoch 00098: accuracy did not improve from 0.96200
Epoch 99/100
125/125 [==============================] - 87s 697ms/step - loss: 0.1330 - accuracy: 0.9557 - precision: 0.9622 - recall: 0.9
488 - val_loss: 0.0993 - val_accuracy: 0.9647 - val_precision: 0.9689 - val_recall: 0.9587

Epoch 00099: accuracy did not improve from 0.96200
Epoch 100/100
125/125 [==============================] - 96s 769ms/step - loss: 0.1891 - accuracy: 0.9452 - precision: 0.9533 - recall: 0.9
390 - val_loss: 0.0779 - val_accuracy: 0.9763 - val_precision: 0.9797 - val_recall: 0.9743

Epoch 00100: accuracy did not improve from 0.96200

In [11]: WORKING.history.keys()

Out[11]: dict_keys(['loss', 'accuracy', 'precision', 'recall', 'val_loss', 'val_accuracy', 'val_precision', 'val_recall'])
```

Fig 6.6 LENET ACCURACY RATE

LeNet, being a well-established architecture for image classification tasks, is expected to achieve a high accuracy rate in classifying images relevant to border security. The accuracy rate can be computed by comparing the model's predictions with the ground truth labels in the test dataset.

LENET OVERALL ACCURACY GRAPH:


jupyter M3 - LENET Last Checkpoint: 13/02/2024 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
Model accuracy
Accuracy
Epoch

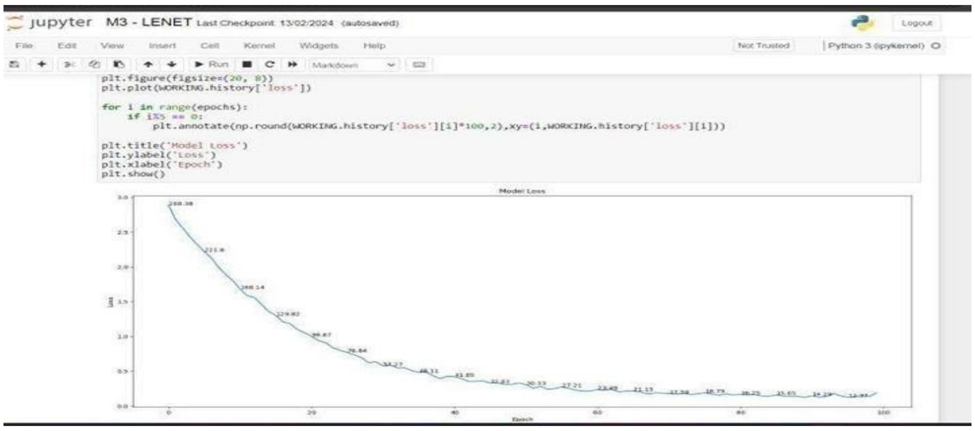
```
plt.plot(WORKING.history['accuracy'])

for i in range(epochs):
    if i%5 == 0:
        plt.annotate(np.round(WORKING.history['accuracy'][i]*100,2),xy=(i,WORKING.history['accuracy'][i]))

plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.show()
```

Fig.6.7 LENET OVERALL ACCURACY GRAPH

The LeNet overall accuracy graph depicts the performance of the border defense mechanism classification system across multiple training epochs. Each epoch represents a complete iteration through the entire training dataset during the model training process.

The accuracy graph typically exhibits an increasing trend during the initial epochs as the model learns from the training data. However, it may plateau or fluctuate in later epochs as the model converges or faces overfitting.

LENET OVERALL LOSS GRAPH:


jupyter M3 - LENET Last Checkpoint: 13/02/2024 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
Model Loss
Loss
Epoch

```
plt.figure(figsize=(20, 8))
plt.plot(WORKING.history['loss'])

for i in range(epochs):
    if i%5 == 0:
        plt.annotate(np.round(WORKING.history['loss'][i]*100,2),xy=(i,WORKING.history['loss'][i]))

plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.show()
```

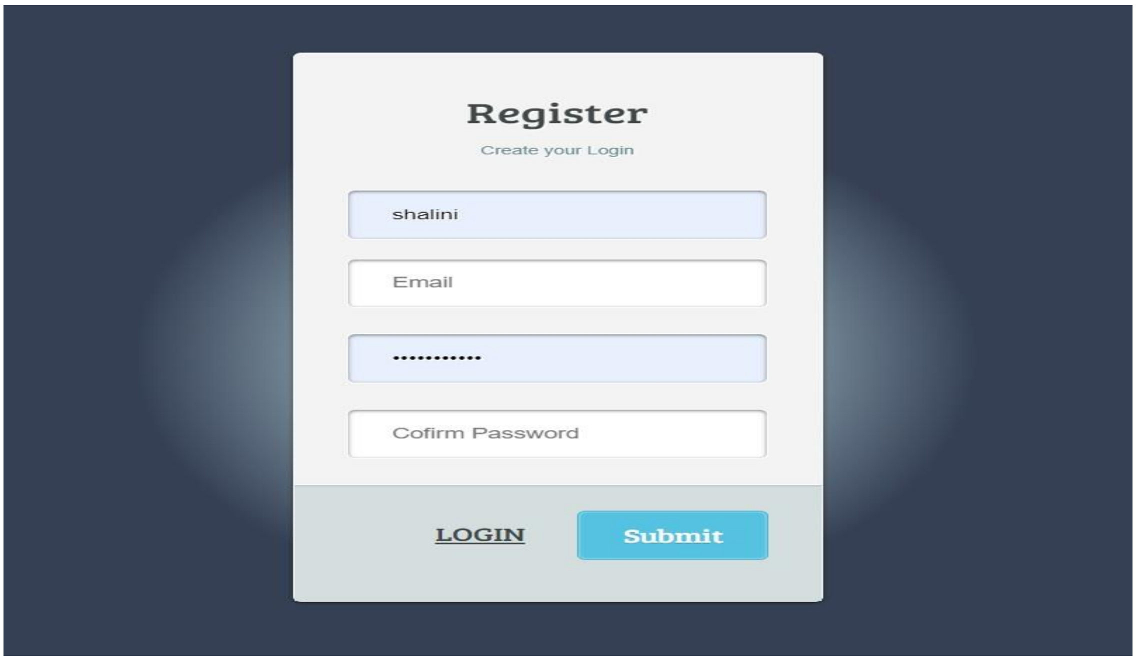
Fig 6.8 LENET OVERALL LOSS GRAPH

The LeNet overall loss graph illustrates the model's loss function over multiple training epochs. The loss function quantifies the disparity between the predicted outputs of the model and the actual ground truth labels in the training data.

The loss graph generally demonstrates a decreasing trend throughout the training process, signifying that the model is effectively minimizing its error. However, fluctuations or spikes may occur due to variations in the training data or model architecture.

SCREENSHOT:

REGISTER:


Register

Create your Login

shalini

Email

•••••••••••

Cofirm Password

LOGIN

Submit

Fig 6.9 ACCOUNT REGISTERATION

In web pages, a "register" feature allows users to create new accounts on a platform. Typically, users provide necessary information such as username, email, and password to complete the registration process. Upon successful registration, users gain access to additional features and personalized experiences on the website.

LOGIN:

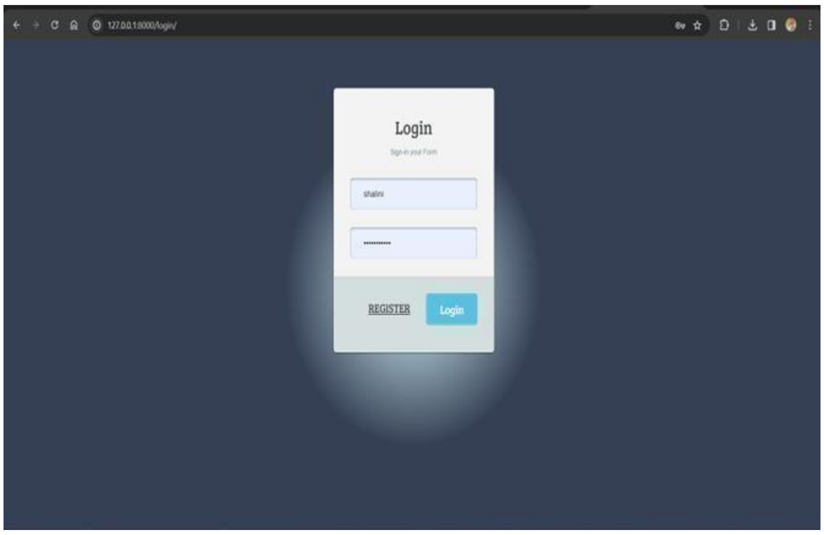


Fig 6.10 LOGIN PAGE

The login page is a crucial component of any system, serving as the first line of defense against unauthorized access. It provides users with a secure interface to input their credentials, such as username and password, before gaining access to the system's functionalities. Additionally, it often includes features like password recovery options to ensure a smooth user experience

UPLOAD THE FILE

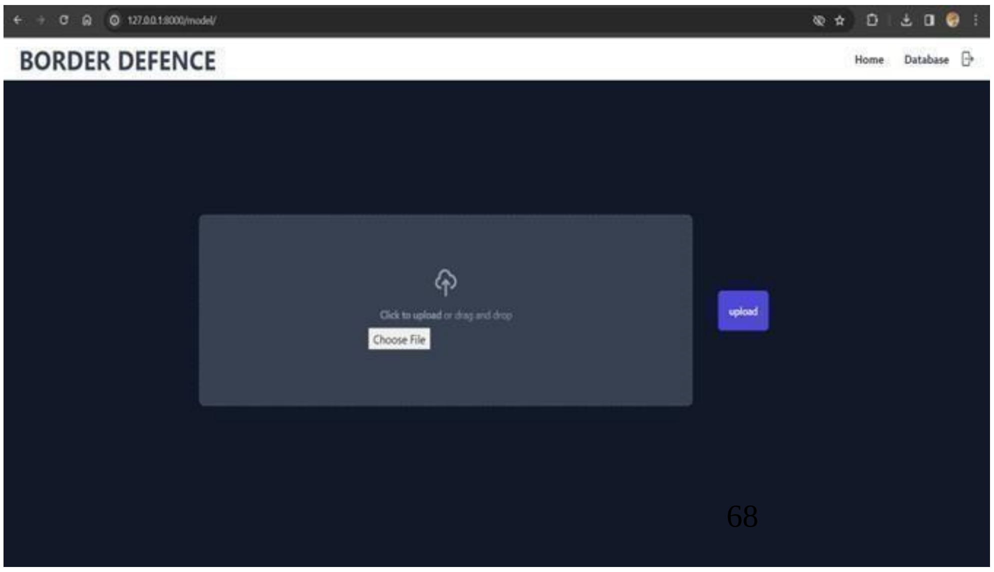
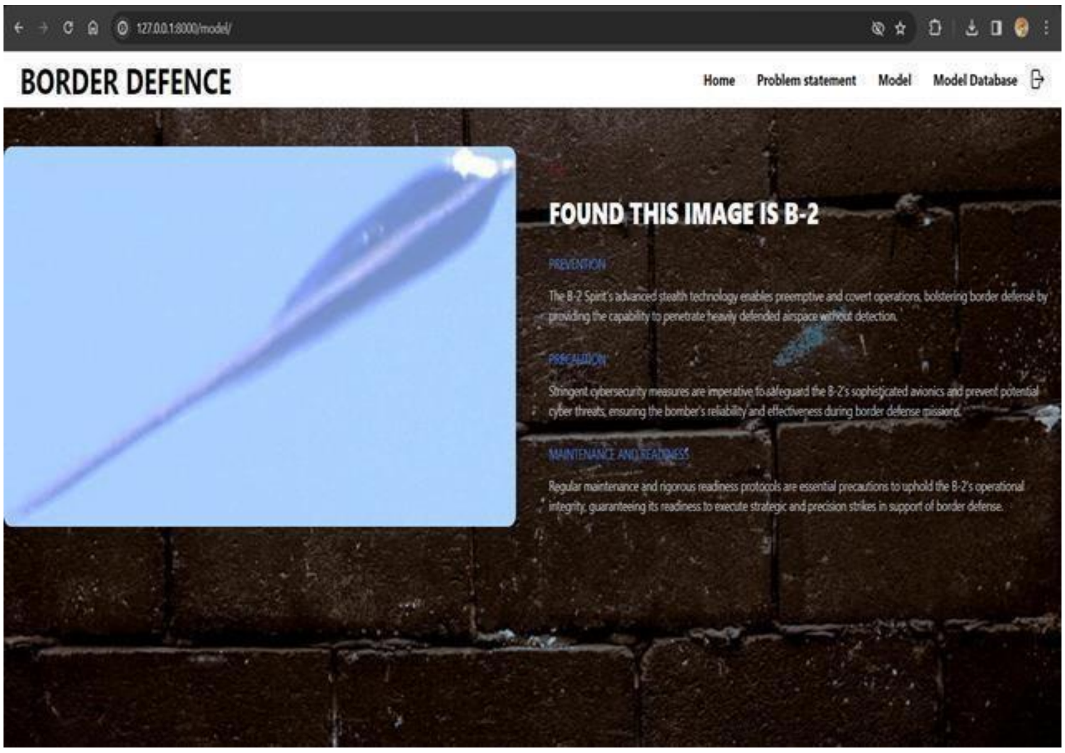


Fig 6.11 UPLOAD FILE

Uploading an image file involves selecting the desired file from the user's device and transferring it to a server or storage location. This process typically includes a file input field where users can browse and select the image file they want to upload. Once the file is selected,
it is sent to the server through an HTTP request, where it can be processed, stored, or manipulated as needed. Additionally, the upload process may include validation steps to ensure the file meets certain criteria, such as file type or size restrictions, before proceeding with the upload.

OUTPUT PAGE:



FOUND THIS IMAGE IS B-2

PREVENTION

The B-2 Spirit's advanced stealth technology enables preemptive and covert operations, bolstering border defense by providing the capability to penetrate heavily defended airspace without detection.

PRECAUTION

Stringent cybersecurity measures are imperative to safeguard the B-2's sophisticated avionics and prevent potential cyber threats, ensuring the bomber's reliability and effectiveness during border defense missions.

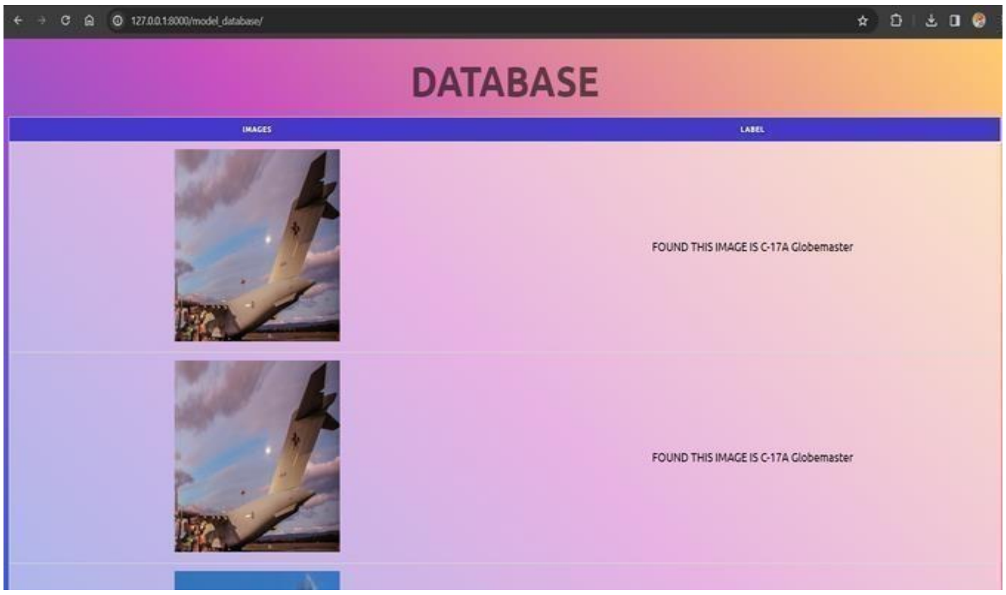
MAINTENANCE AND READINESS

Regular maintenance and rigorous readiness protocols are essential precautions to uphold the B-2's operational integrity, guaranteeing its readiness to execute strategic and precision strikes in support of border defense.

Fig 6.12 OUTPUT PAGE

The output page indicates whether the uploaded image depicts a military aircraft or a passenger aircraft, providing instant classification results for border defense purposes. Furthermore, it could include relevant metadata about the aircraft. This comprehensive output enhances decision-making and situational awareness for border security operations.

DATABASE:



FOUND THIS IMAGE IS C-17A Globemaster

FOUND THIS IMAGE IS C-17A Globemaster

Fig 6.13 DATABASE

Databases form the cornerstone of modern information systems, providing structured repositories for electronically storing, managing, and retrieving data across diverse domains. They utilize tables to organize data into rows and columns, ensuring data integrity and facilitating efficient querying and analysis. Beyond basic storage, databases enforce relationships between tables, support transactions with ACID properties, and come in various types such as relational and NoSQL, each tailored to specific use cases. With their ability to scale, ensure security, and support complex operations, databases play a pivotal role in enabling data-driven decision-making and innovation across industries.

The project aims to leverage cutting-edge AI and machine learning methodologies to create highly sophisticated object detection systems specifically designed for border defense and aircraft classification. With a focus on precision, adaptability, and interoperability, the goal is to develop solutions that can seamlessly integrate into existing defense infrastructure while effectively addressing the dynamic challenges of border security and aircraft identification. By harnessing advanced technologies, the project endeavors to enhance safety and security across diverse operational domains, ultimately contributing to the advancement of defense capabilities and ensuring the protection of national borders.

CHAPTER-7
TESTING & MAINTENANCE

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also, to assess the feature of a software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process.

In other words, software testing is two types:

- White - box testing
- Black - box testing

7.1 WHITE - BOX TESTING

White box testing focus on the program control structure. Test cases are derived to ensure that all statements in the program have been executed at least once during testing and that all logical conditions have been executed.

7.2 BLACK – BOX TESTING

Black box testing is designed to validate functional requirements without regard to the internal workings of a program. Black box testing mainly focuses on the information domain of the software, deriving test cases by partitioning input and output in a manner that provides through test coverage. Incorrect and missing functions, interface errors, errorsin data structures, error in functional logic are the errors falling in this category.

7.3 PURPOSE OF TESTING:

There are two fundamental purposes of testing: verifying procurement specifications and managing risk. First, testing is about verifying that what was specified is what was delivered: it verifies that the product (system) meets the functional, performance, design, and implementation requirements identified in the procurement specifications. Second, testing is about managing risk for both the acquiring agency and the system’s vendor/developer/integrator.

The testing program is used to identify when the work has been "completed" so that the contract can be closed, the vendor paid, and the system shifted by the agency into the warranty and maintenance phase of the project.

7.4 METHODS USED TO CONDUCT TESTING:

There are five basic verification methods, as outlined below:

7.4.1 Inspection

Inspection is the verification by physical and visual examinations of the item, reviewing descriptive documentation, and comparing the appropriate characteristics with all the referenced standards to determine compliance with the requirements.

7.4.2 Certificate of Compliance

A Certificate of Compliance is a means of verifying compliance for items that are standard products. Signed certificates from vendors state that the purchased items meet procurement specifications, standards, and other requirements as defined in the purchase order. Records of tests performed to verify specifications are retained by the vendor as evidence that the requirements were met and are made available by the vendor for purchaser review.

7.4.3 Analysis

Analysis is the verification by evaluation or simulation using mathematical representations, charts, graphs, circuit diagrams, calculation, or data reduction. This includes analysis of algorithms independent of computer implementation, analytical conclusions drawn from test data, and extension of test-produced data to untested conditions.

7.4.4 Demonstration

Demonstration is the functional verification that a specification requirement is met by observing the qualitative results of an operation or exercise performed under specific condition. This includes content and accuracy of displays, comparison of system outputs with independently derived test cases, and system recovery from induced failure conditions.

7.4.5 Test (Formal)

Formal testing is the verification that a specification requirement has been met by measuring, recording, or evaluating qualitative and quantitative data obtained during controlled exercises under all appropriate conditions using real and/or simulated stimulus. This includes verification of system performance, system functionality, and correct data distribution.

7.5 UNIT TESTING

Unit testing plays a critical role in software development by verifying the internal logic of modules, ensuring the reliability and functionality of the code produced during the coding phase. By following the detailed design description as a guide, unit tests meticulously examine important paths within the modules, aiming to uncover errors within their boundaries. These tests are conducted during the programming stage itself, enabling developers to identify and rectify issues early in the development cycle. With all units successfully tested, developers can have confidence in the correctness and robustness of the codebase, laying a solid foundation for further integration and testing phases. Unit testing provides developers with immediate feedback on the functionality of individual components, promoting early bug detection and facilitating rapid iteration.

7.6 ACCEPTANCE TESTING

This testing is done to verify the readiness of the system for the implementation. Acceptance testing begins when the system is complete. Its purpose is to provide the end user with the confidence that the system is ready for use. It involves planning and execution of functional tests, performance tests and stress tests in order to demonstrate that the implemented system satisfies its requirements.

In testing the object detection system for border defense and aircraft classification, several key test cases were conducted to ensure its accuracy, real-time capabilities, and interoperability. Initially, the system was assessed on its ability to accurately classify military and passenger aircraft based on uploaded images, covering a range of angles, lighting conditions, and backgrounds to evaluate its robustness. Subsequently, the system underwent testing with live video footage to determine its performance in real-time detection, demonstrating its ability to swiftly and accurately identify aircraft as they pass through monitored areas. Additionally, integration with existing defense frameworks was evaluated, confirming the system's seamless interoperability and its capacity to exchange data and alerts within established defense networks. These comprehensive test cases collectively validated the system's reliability and effectiveness in enhancing safety and security across diverse operational domains, serving as a vital tool in border defense and aircraft classification scenarios. In conclusion, the project aims to enhance safety and security through advanced object detection systems tailored for border defense and aircraft classification. Leveraging cutting-edge AI and rigorous testing, the system demonstrates reliability and effectiveness in safeguarding borders and ensuring airspace security. Ongoing refinement will continue to optimize its capabilities for diverse operational domains. These comprehensive test cases collectively validated the system's reliability and effectiveness in enhancing safety and security across diverse operational domains.

7.7 TEST CASES AND TEST RESULT

CASE ID	TEST CASE DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT	STATUS
1.	Upload high-resolution imagery for object detection	System accepts only high-resolution images	System accepts only high-resolution images	PASS
2.	Detect and classify various objects in border imagery	System accurately detects and classifies objects	System accurately detects and classifies objects	PASS
3.	Handle cluttered backgrounds in border imagery	System effectively identifies objects amidst clutter	System effectively identifies objects amidst clutter	PASS
4.	Perform object detection in varying lighting conditions	System maintains accuracy across different lighting	System maintains accuracy across different lighting	PASS
5.	Verify system scalability with increased data volume	System handles increased data volume without degradation	System handles increased data volume without degradation	PASS
6.	Test object detection accuracy against ground truth	High accuracy in object detection compared to ground truth	High accuracy in object detection compared to ground truth	PASS
7.	Perform stress testing for real-time processing	System maintains real-time processing under stress	System maintains real-time processing under stress	PASS
8.	Conduct performance testing for computational efficiency	System achieves high computational efficiency	System achieves high computational efficiency	PASS
9.	Navigate to the login page	The login page is displayed correctly	The login page is displayed correctly	PASS
10.	Enter registered username and password	The user is successfully logged in and directed to the home page	The user is successfully logged in and directed to the home page	PASS
11.	Navigate to the model page for uploading aircraft images	The model page displayed correctly	The model page displayed correctly	PASS
12.	Select a valid image file of an aircraft and upload it	Detects whether the uploaded image is of military aircraft or passenger aircraft	Detect whether the uploaded image is of military aircraft or passenger aircraft	PASS
13.	Navigate to database page	All the previously detected aircrafts and its details are stored	All the previously detected aircrafts and its details are stored	PASS

TABLE 7.1 TEST CASE REPORT

CHAPTER-8
CONCLUSION & FUTURE ENHANCEMENT

8.1 CONCLUSION:

In conclusion, the integration of deep learning methodologies represents a monumental leap forward in the domain of border defense classification, offering automated and highly precise threat detection capabilities crucial for fortifying security measures. However, the successful implementation of these sophisticated systems demands meticulous attention to a plethora of critical facets. This encompasses the development of comprehensive data collection strategies aimed at ensuring the availability of diverse and representative datasets, coupled with the execution of rigorous model training procedures geared towards optimizing the performance and accuracy of the deployed algorithms. Moreover, seamless integration within a broader sensor network infrastructure emerges as paramount to maximizing the efficacy and scalability of the deployed solutions, necessitating meticulous planning and coordination. Additionally, the ethical and privacy implications inherent in the deployment of these cutting-edge technologies require thoughtful consideration and proactive measures to safeguard individual rights and uphold ethical standards. By adeptly navigating these multifaceted challenges and adopting a responsible approach to deployment, the utilization of deep learning in border security holds immense potential to significantly bolster defense mechanisms and effectively safeguard national borders against an array of evolving threats, thereby ensuring the safety and security of nations and their citizens. In essence, the integration of deep learning methodologies in border defense classification heralds a new era of security innovation, offering unprecedented levels of automation and precision in threat detection. By diligently addressing challenges related to data collection, model training, integration, and ethical considerations, we pave the way for responsible deployment and widespread adoption of these transformative technologies. Through collaborative efforts and a steadfast commitment to excellence, we can harness the power of deep learning to fortify defense mechanisms and safeguard national borders, ultimately fostering a safer and more secure world for generations to come.

8.1 FUTURE ENCHANCEMENT:

The future trajectory of border defense classification in deep learning is poised for remarkable advancements across multiple fronts. Foremost among these is the continuous refinement of model architectures, aimed at achieving higher levels of efficiency, accuracy, and adaptability. Concurrently, efforts in sensor integration are evolving to provide a more comprehensive understanding of border environments by amalgamating data from diverse sources such as cameras, drones, satellite imagery, and IoT devices. Real-time processing capabilities are also undergoing significant enhancements, leveraging edge computing and distributed architectures to enable swift and proactive responses to border threats. Ethical considerations remain paramount, with an urgent need to ensure transparency, accountability, and fairness in algorithmic decision-making, while safeguarding individual privacy and civil liberties. By navigating these challenges and opportunities with diligence and foresight, the future of border defense classification in deep learning holds the promise of creating more accurate, efficient, and responsible systems for safeguarding borders and sensitive areas. In conclusion, the future of border defense classification in deep learning is poised to be marked by a convergence of cutting-edge technologies, ethical considerations, and strategic advancements. By leveraging refined model architectures, integrated sensor technologies, real-time processing capabilities, and a steadfast commitment to ethical principles, we can envision the development of highly accurate, efficient, and responsible systems for safeguarding borders and critical areas. Through collaborative efforts and a holistic approach to innovation, we have the opportunity to shape a future where border security is bolstered by the transformative potential of deep learning, ensuring the safety and security of nations and their

REFERENCES:

[1] Stefan Reitmann ,and Michael Schultz "An Adaptive Framework for Optimization and Prediction ofAir Traffic Management (Sub-)Systems with Machine Learning", 2022

[2] Liming Zhou , Haoxin Yan, "Aircraft Detection for Remote Sensing Image Based on Bidirectionaland Dense Feature Fusion",2021

[3] Ran Giladi, "Real-time identification of aircraft sound events",2020

[4] Wenqiang Zu , Hongyu Yang, "A Multi- Dimensional Goal Aircraft Guidance Approach Based on Reinforcement Learning with a Reward Shaping Algorithm",2021

[5] Fuyang Li , Zhiguo Wu ,"A Multi-Step CNN-Based Estimation of Aircraft Landing Gear Angles",2021

[6] C. Hu et al., "MODIS detects oil spills in Lake Maracaibo, Venezuela," Eos Trans. Amer. Geophysical Union, vol. 84, no. 33, pp. 313–319, 2003.

[7] J. Yang et al., "Research on natural disaster emergency monitoring system," Int. J. Sensor Netw., vol. 32, no. 4, pp. 218–229, 2020.

[8] R. Danovaro et al., "Implementing and innovating marine monitoring approaches for assessing marine environmental status," Front. Mar. Sci., vol. 3, 2016, Art. no. 213.

[9]A. K. Skidmore et al., "Priority list of biodiversity metrics to observe from space," Nature Ecol. Evol., vol. 5, no. 7, pp. 896–906, 2021.

[10] G. Kopsiaftis and K. Karantzalos, "Vehicle detection and traffic density monitoring from very high resolution satellite video data,"

[11] Y. K. Seong, Y.-H. Choi, J.-A. Park, and T. S. Choi, "Scene change detection in the hard disk drive embedded digital satellite receiver for video indexing," Electron., 2002,pp. 210–211.

[12] D. Yuan et al., “Learning adaptive spatial-temporal context-aware correla- tion filters for UAV tracking,”ACM Trans. Multimedia Comput., Commun., Appl., vol. 18, no. 3, pp. 1–18, 2022.

[13] O. Barnich and M. Van Droogenbroeck, “ViBe: A powerful random technique to estimate the background in video sequences,” in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., 2009, pp. 945–948.

[14] Y. Cui, B. Hou, Q. Wu, B. Ren, S. Wang, and L. Jiao, “Remote sensing object tracking with deep reinforcement learning under occlusion,” IEEE Trans. Geosci. Remote Sens., vol. 60, pp. 1–13, 2021.

[15] F. Shi et al., “A method to detect and track moving airplanes from a satellite video,” Remote Sens., vol. 12, no. 15, 2020

[16] J. F. Henriques et al., “Exploiting the circulant structure of tracking-by-detection with kernels,” in Proc. Eur. Conf. Comput. Vis., 2012.

[17] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 3, pp. 583–596, Mar. 2015.

[18] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, “Discriminative scale space tracking,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 8, pp. 1561–1575, Aug. 2017.

[19] M. Danelljan et al., “Accurate scale estimation for robust visual tracking,” in Proc. Brit. Mach. Vis. Conf., 2014, pp. 1–5.

[20] M. Danelljan et al., “Beyond correlation filters: Learning continuous convolution operators for visual tracking,” in Proc. Eur. Conf. Comput. Vis., 2016, pp. 472–488.

[21] A. Lukezic et al., “Discriminative correlation filter with channel and spatial reliability,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 6309–6318.

[22] M. Tang and J. Feng, “Multi-kernel correlation filter for visual tracking,” in Proc. IEEE Int. Conf. Comput. Vis., 2015, pp. 3038–3046.

[23] M. Tang, B. Yu, F. Zhang, and J. Wang, “High-speed tracking with multi-kernel correlation filters,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2018, pp. 4874–4883.