

**NAZI INFANTRY AND ARTILLERY WEAPONS
CLASSIFICATIONS USING NEURAL NETWORK
TECHNIQUES**

A PROJECT REPORT

Submitted by

GURU VIDHYA S (411620243009)

SARANYA R P (411620243032)

In partial fulfilment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE



**PRINCE Dr.K. VASUDEVAN COLLEGE OF
ENGINEERING AND TECHNOLOGY,
PONMAR, CHENNAI-600 127**

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2024

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report “**NAZI INFANTRY AND ARTILERRY WEAPONS CLASSIFICATION USING NEURAL NETWORK TECHNIQUES**” is the bonafide of “**GURUVIDHYA S (411620243009)**” and “**SARANYA R P(411620243032)**”, who carried out the project work under my supervision.

SIGNATURE

Mrs. B.UMA MAHESHWARI, M.TECH.,

HEAD OF DEPARTMENT

Department of AI & DS,
Prince Dr. K. Vasudevan College
of Engineering and Technology,
Chennai-600127

SIGNATURE

Mrs. M. VANITHA, M.E,

SUPERVISOR

Department of AI & DS,
Prince Dr. K. Vasudevan college
of Engineering and Technology,
Chennai-6000127

Submitted to the Viva voce Examination held on _ _ _ _ _

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We wish to express our sincere thanks to our **FOUNDER AND CHAIRMAN, Dr. K. VASUDEVAN, M.A., B.Ed., Ph.D.**, for his endeavour in educating us in his premier institution.

We would like to extend our heartfelt gratitude and sincere thanks to our **VICE CHAIRMAN, Dr. V. VISHNU KARTHIK, M.D.**, for his keen interest in our studies and the facilities offered in this premier institution.

We would like to express our deep gratitude and sincere thanks to our **ADMINISTRATIVE OFFICER, Dr. K. PARTHASARATHY BE.**, for his valuable support.

We wish to express our sincere thanks to our **HONOURABLE PRINCIPAL, Dr. T. SUNDER SELWYN, M.E., Ph.D.**, for permitting to access various resources in the college to complete the project work

We also wish to convey our thanks and regards to our **HOD** and Project Co-ordinator, **Mrs. B. UMA MAHESHWARI, M.TECH.**, Department of Artificial Intelligence and Data Science, for her guidance and support throughout our project.

We wish to express our great deal of gratitude to our Project Guide , **Mrs. M. VANITHA, M.E.**, Department of Artificial Intelligence and Data Science for the pleasure guidance to finish our project successfully.

We would like to extend our thanks to all teaching and non-teaching staffs of Department of Artificial Intelligence and Data Science for their continuous support.

ABSTRACT

Neural network techniques are harnessed to systematically classify Nazi infantry and artillery weaponry based on their distinctive attributes. By leveraging the power of neural networks, a robust system is constructed for automated classification. This involves the extraction of intricate features and patterns inherent to each weapon category, enabling precise differentiation. The neural network's architecture is tailored to accommodate the complexity of the weapons dataset, facilitating accurate recognition and classification. Through this innovative approach, a comprehensive understanding of Nazi weaponry taxonomy is achieved, contributing to historical analysis and military studies. The utilization of neural networks in this context exemplifies their potential in handling complex categorization tasks, bridging the gap between advanced technology and historical research.

Keywords: Nazi, infantry, artillery, weapons, classifications, neural network techniques, automated classification, distinctive attributes, patterns, taxonomy, historical analysis, military studies, technology.

TABLE OF CONTENTS

| CHAPTER NO | TITLE | PAGE NO |
|------------|---|-----------|
| | ABSTRACT | I |
| | LIST OF FIGURES | IV |
| | LIST OF ABBREVIATIONS | V |
| 1 | INTRODUCTION | 1 |
| | 1.1. Domain Introduction | 1 |
| | 1.2. Introduction | 1 |
| | 1.3. Project Definition | 2 |
| | 1.3. Project Description | 3 |
| 2 | LITERATURE SURVEY | 4 |
| 3 | SYSTEM ANALYSIS | 15 |
| | 3.1. Existing System | 15 |
| | 3.1.1. Disadvantages | 16 |
| | 3.2. Proposed System | 16 |
| | 3.2.1. Advantages | 16 |
| | 3.3. Feasibility Study | 17 |
| | 3.5.1 Splitting the dataset | 17 |
| | 3.5.2 Construction of a Detecting Model | 17 |
| 4 | SYSTEM DESIGN | 18 |
| | 4.1. System Architecture | 18 |
| | 4.2. UML Diagrams | 19 |
| | 4.2.1. Data Flow Diagram | 19 |
| | 4.2.2. Work Flow Diagram | 20 |
| | 4.2.3. Use Case Diagram | 21 |

| | | |
|----------|--|-----------|
| | 4.2.4. Class Diagram | 22 |
| | 4.2.5. Activity Diagram | 23 |
| | 4.2.6. Sequence Diagram | 24 |
| | 4.2.7. ER Diagram | 25 |
| | 4.2.8. Collaboration Diagram | 26 |
| 5 | SYSTEM REQUIREMENTS | 27 |
| | 5.1. Hardware Requirements | 27 |
| | 5.2. Software Requirements | 27 |
| 6 | SYSTEM IMPLEMENTATION | 28 |
| | 6.1. Software Description | 28 |
| | 6.1.1. Anaconda Navigator | 28 |
| | 6.1.2. Jupyter Notebook | 30 |
| | 6.1.3. Python | 34 |
| | 6.2. List of Modules | 37 |
| | 6.3. Module Description | 38 |
| | 6.3.1. Import the given image from dataset | 38 |
| | 6.3.2. To train the module by image dataset | 38 |
| | 6.3.3. Working process of layers in CNN model | 39 |
| | 6.3.4. Pneumonia classification identification | 41 |
| 7 | DEPLOYMENT | |
| | 7.1. Django(Web Framework) | 42 |
| | 7.2. Working Process | 44 |
| 8 | CODING AND RESULT | 45 |
| 9 | CONCLUSION AND FUTURE ENHANCEMENT | 53 |
| | REFERENCES | 55 |

LIST OF FIGURES

| FIGURE NO | NAME OF THE FIGURE | PAGE NO |
|------------------|--|----------------|
| 4.1 | System Architecture | 18 |
| 4.2 | Data Flow Diagram | 19 |
| 4.3 | Work Flow Diagram | 20 |
| 4.4 | Use Case diagram | 21 |
| 4.5 | Class Diagram | 22 |
| 4.6 | Activity Diagram | 23 |
| 4.7 | Sequence Diagram | 24 |
| 4.8 | ER Diagram | 25 |
| 4.9 | Collaboration Diagram | 26 |
| 6.1 | Import the given image from dataset | 37 |
| 6.2 | To train the module by given image dataset | 38 |
| 6.3 | Pneumonia classification identification | 39 |

LIST OF ABBREVIATIONS

ACRONYMS

DESCRIPTION

| | |
|------|---|
| CNN | Convolutional Neural Network |
| RNN | Recurrent Neural Network |
| SOD | Salient Object Detection |
| ICON | Integrity Cognition Network |
| DETR | Detection Transformer |
| TQE | Temporal Query Encoder |
| TDTD | Temporal Deformable Transformer Decoder |
| GCNs | Graph Convolutional Networks |
| GNNs | Graph Neural Networks |
| ICE | Integrity Channel Enhancement |
| DFA | Diverse Feature Aggregation |

CHAPTER 1

INTRODUCTION

1.1 DOMAIN INTRODUCTION

Deep learning is a branch of machine learning which is completely based on artificial neural networks, as neural network is going to mimic the human severity so deep learning is also a kind of mimic of human severity. It's on hype nowadays because earlier we did not have that much processing power and a lot of data. A formal definition of deep learning is- neurons Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones. In severity approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousands of their neighbors. The question here is how it recreates these neurons in a computer. So, it creates an artificial structure called an artificial neural net where we have nodes or neurons. It has some neurons for input value and some for output value and in between, there may be lots of neurons interconnected in the hidden layer.

1.2 INTRODUCTION

The study of historical military artifacts, particularly weapons used during World War II, holds significant importance for understanding the technological advancements and strategies employed by different nations. Among the various factions involved, the weaponry of the Nazi regime, characterized by its innovation and engineering prowess, stands out as a subject of intense interest.

This research aims to employ neural network techniques for the systematic classification of Nazi infantry and artillery weapons used during World War II. By

leveraging the capabilities of artificial intelligence, we seek to enhance the accuracy and efficiency of weapon identification, enabling a deeper exploration of historical contexts, manufacturing processes, and tactical considerations associated with these instruments of war.

The study involves the development and training of a neural network model using a curated dataset of images depicting various Nazi infantry rifles, submachine guns, pistols, and artillery pieces. This dataset encompasses a diverse range of models and variants, capturing the nuances in design that evolved over the course of the war. The neural network will be trained to recognize distinct features, such as barrel length, stock design, and overall configuration, allowing it to differentiate between different classes of weapons.

By applying neural network techniques, we aim to achieve a level of accuracy and efficiency in weapon classification that surpasses traditional methods. This will facilitate the creation of a comprehensive database, providing a valuable resource for military historians, collectors, and educators. The outcomes of this research not only contribute to our understanding of World War II weaponry but also showcase the potential of artificial intelligence in historical analysis.

1.3 PROBLEM DEFINITION

Using neural network techniques for the classification of Nazi infantry and artillery weapons involves leveraging advanced machine learning algorithms to categorize and identify various firearms and artillery pieces utilized by the Nazi forces during World War II. The neural network is trained on a dataset comprising images, specifications, and historical information related to these weapons. The model learns to recognize distinctive features such as gun barrels, stocks, and other components, enabling it to classify weapons accurately. This approach enhances the efficiency of cataloging and analyzing historical military equipment, providing a valuable tool for researchers, historians, and enthusiasts to gain insights into the

technological aspects of Nazi weaponry and its impact on the course of history. It is important to approach such studies with a focus on historical understanding and education rather than glorification or promotion of ideologies associated with the Nazi regime.

1.4 PROJECT DESCRIPTION

The proposed system aims to employ advanced neural network techniques for the classification of Nazi infantry and artillery weapons. Leveraging deep learning architectures, the system will analyze historical images, schematics, and textual descriptions to accurately categorize and label these weapons. The neural network will be trained on a comprehensive dataset comprising diverse examples of Nazi weaponry. Through convolutional neural networks (CNNs) for image analysis and recurrent neural networks (RNNs) for text processing, the system will learn intricate features and relationships within the data. This hybrid approach will enable precise classification, distinguishing between various types of rifles, handguns, machine guns, and artillery pieces. By harnessing the power of neural networks, the proposed system seeks to enhance our understanding of these historical artifacts while contributing to the preservation of historical knowledge.

CHAPTER 2

LITERATURE SURVEY

TITLE : SALIENT OBJECT DETECTION VIA INTEGRITY LEARNING

AUTHOR: Mingchen Zhuge , deng-ping , Dong Xu , Nian Liu , Dingwen Zhang

YEAR : 2023

DESCRIPTION: Although current salient object detection (SOD) works have achieved significant progress, they are limited when it comes to the integrity of the predicted salient regions. We define the concept of integrity at both a micro and macro level. Specifically, at the micro level, the model should highlight all parts that belong to a certain salient object. Meanwhile, at the macro level, the model needs to discover all salient objects in a given image. To facilitate integrity learning for SOD, we design a novel Integrity Cognition Network (ICON), which explores three important components for learning strong integrity features.

Salient object detection (SOD) aims to imitate the human visual perception system to capture the most significant regions. Performance metrics such as precision, recall, F-measure, and mean absolute error (MAE) are used to assess the effectiveness of the integrity learning approach in accurately detecting salient objects. Salient Object Detection via Integrity Learning represents an innovative approach that integrates integrity learning principles into the salient object detection process, enabling the model to capture not only saliency but also the structural coherence and relationships within visual scenes, leading to more accurate and contextually meaningful salient object detection results.

TITLE : CLASSIFICATION OF DIFFERENT WEAPON TYPES USING DEEP LEARNING

AUTHOR: Volkan Kaya 1,*, Servet Tuncer 2 and Ahmet Baran 1

YEAR : 2023

DESCRIPTION: Today, with the increasing number of criminal activities, automatic control systems are becoming the primary need for security forces. In this study, a new model is proposed to detect seven different weapon types using the deep learning method. This model offers a new approach to weapon classification based on the VGGNet architecture. The model is taught how to recognize assault rifles, bazookas, grenades, hunting rifles, knives, pistols, and revolvers. The proposed model is developed using the Keras library on the TensorFlow base. A new model is used to determine the method required to train, create layers, implement the training process, save training in the computer environment, determine the success rate of the training, and test the trained model. In order to train the model network proposed in this study, a new dataset consisting of seven different weapon types is constructed. Using this dataset, the proposed model is compared with the VGG-16, ResNet-50, and ResNet-101 models to determine which provides the best classification results.

As a result of the comparison, the proposed model's success accuracy of 98.40% is shown to be higher than the VGG-16 model with 89.75% success accuracy, the ResNet-50 model with 93.70% success accuracy, and the ResNet-101 model with 83.33% success accuracy.

TITLE : DENOISED SELF-TRAINING FOR UNSUPERVISED DOMAIN ADAPTATION ON 3D OBJECT DETECTION

AUTHOR: Jihan Yang, Shaoshuai Shi , Zhe Wang and Hongsheng Li

YEAR : 2023

DESCRIPTION: In this paper, we present a self-training method, named ST3D++, with a holistic pseudo label denoising pipeline for unsupervised domain adaptation on 3D object detection. ST3D++ aims at reducing noise in pseudo label generation as well as alleviating the negative impacts of noisy pseudo labels on model training. First, ST3D++ pre-trains the 3D object detector on the labeled source domain with random object scaling (ROS) which is designed to reduce target domain pseudo label noise arising from object scale bias of the source domain. Then, the detector is progressively improved through alternating between generating pseudo labels and training the object detector with pseudo-labeled target domain data.

Here, we equip the pseudo label generation process with a hybrid quality-aware triplet memory to improve the quality and stability of generated pseudo labels. Meanwhile, in the model training stage, we propose a source data assisted training strategy and a curriculum data augmentation policy to effectively rectify noisy gradient directions and avoid model over-fitting to noisy pseudo labeled data. These specific designs enable the detector to be trained on meticulously refined pseudo labeled target data with denoised training signals, and thus effectively facilitate adapting an object detector to a target domain without requiring annotations.

TITLE : TRANSVOD: END-TO-END VIDEO OBJECT DETECTION WITH SPATIAL-TEMPORAL TRANSFORMERS

AUTHOR: Qianyu Zhou , Xiangtai Li , Lu He, Yibo Yang , Guangliang Cheng

YEAR : 2023

DESCRIPTION: Detection Transformer (DETR) and Deformable DETR have been proposed to eliminate the need for many hand-designed components in object detection while demonstrating good performance as previous complex hand-crafted detectors. However, their performance on Video Object Detection (VOD) has not been well explored. In this paper, we present TransVOD, the first end-to-end video object detection system based on simple yet effective spatial-temporal Transformer architectures. The first goal of this paper is to streamline the pipeline of current VOD, effectively removing the need for many hand-crafted components for feature aggregation, e.g., optical flow model, relation networks. Besides, benefited from the object query design in DETR, our method does not need postprocessing methods such as Seq-NMS. In particular, we present a temporal Transformer to aggregate both the spatial object queries and the feature memories of each frame.

Our temporal transformer consists of two components: Temporal Query Encoder (TQE) to fuse object queries, and Temporal Deformable Transformer Decoder (TDTD) to obtain current frame detection results. These designs boost the strong baseline deformable DETR by a significant margin (3 %-4 % mAP) on the ImageNet VID dataset. TransVOD yields comparable performances on the benchmark of ImageNet VID. Then, we present two improved versions of TransVOD including TransVOD++ and TransVOD Lite.

TITLE : WHEN OBJECT DETECTION MEETS KNOWLEDGE DISTILLATION: A SURVEY

AUTHOR : João Ricardo Afonso Pires , Victor Gomes Lauriano Souza

YEAR : 2023

DESCRIPTION: Object detection (OD) is a crucial computer vision task that has seen the development of many algorithms and models over the years. While the performance of current OD models has improved, they have also become more complex, making them impractical for industry applications due to their large parameter size. To tackle this problem, knowledge distillation (KD) technology was proposed in 2015 for image classification and subsequently extended to other visual tasks due to its ability to transfer knowledge learned by complex teacher models to lightweight student models. This paper presents a comprehensive survey of KD-based OD models developed in recent years, with the aim of providing researchers with an overview of recent progress in the field. We conduct an in-depth analysis of existing works, highlighting their advantages and limitations, and explore future research directions to inspire the design of models for related tasks.

We summarize the basic principles of designing KD-based OD models, describe related KD-based OD tasks, including performance improvements for lightweight models, catastrophic forgetting in incremental OD, small object detection, and weakly/semi-supervised OD. We also analyze novel distillation techniques. Additionally, we provide an overview of the extended applications of KD-based OD models on specific datasets, such as remote sensing images and 3D point cloud datasets. We compare and analyze the performance of different models on several common datasets and discuss promising directions for solving specific OD problems.

TITLE : CYCLE CONSISTENCY DRIVEN OBJECT DISCOVERY

AUTHOR: Aniket Didolkar, Anirudh Goyal, Yoshua Bengio

YEAR : 2023

DESCRIPTION: Developing deep learning models that effectively learn object-centric representations, akin to human cognition, remains a challenging task. Existing approaches facilitate object discovery by representing objects as fixed-size vectors, called "slots" or "object files". While these approaches have shown promise in certain scenarios, they still exhibit certain limitations. First, they rely on architectural priors which can be unreliable and usually require meticulous engineering to identify the correct objects. Second, there has been a notable gap in investigating the practical utility of these representations in downstream tasks. To address the first limitation, we introduce a method that explicitly optimizes the constraint that each object in a scene should be associated with a distinct slot. We formalize this constraint by introducing consistency objectives which are cyclic in nature.

To tackle the second limitation, we apply the learned object-centric representations from the proposed method to two downstream reinforcement learning tasks, demonstrating considerable performance enhancements compared to conventional slot-based and monolithic representation learning methods. Our results suggest that the proposed approach not only improves object discovery, but also provides richer features for downstream tasks.

TITLE : SALIENT OBJECT DETECTION IN THE DEEP LEARNING ERA

AUTHOR: W. Wang, Q. Lai, H. Fu, J. Shen, H. Ling, and R. Yang

YEAR : 2022

DESCRIPTION: Salient object detection in the deep learning era has seen significant advancements and improvements due to the availability of large-scale datasets, powerful computational resources, and innovative deep learning architectures. Salient object detection aims to identify and highlight the most visually distinctive objects or regions in an image, which are likely to attract human attention. Deep convolutional neural networks (CNNs) have become the cornerstone of salient object detection. Various architectures, including fully convolutional networks (FCNs), U-Net, DeepLab, and ResNet, have been adapted and extended for saliency detection tasks. These architectures leverage the hierarchical features learned from data to accurately predict salient regions in images. Efforts have been made to develop efficient and lightweight deep learning models for real-time salient object detection applications.

Domain adaptation and transfer learning techniques enable the adaptation of pre-trained models to new domains or datasets with limited labeled data, thereby improving the generalization and performance of salient object detection models in diverse scenarios. Overall, the deep learning era has witnessed remarkable progress in salient object detection, with state-of-the-art algorithms achieving impressive results on benchmark datasets and demonstrating potential applications in fields like image editing, content-aware image retargeting, and visual attention modeling.

TITLE : RE-THINKING CO-SALIENT OBJECT DETECTION

AUTHOR: D.-P. Fan et al.

YEAR : 2021

DESCRIPTION: Co-salient object detection, also known as co-saliency detection, aims to identify objects that are commonly salient across a set of related images. This task has gained increasing attention due to its relevance in various applications such as image retrieval, video analysis, and object recognition. Leveraging deep learning architectures, especially convolutional neural networks (CNNs), has revolutionized co-salient object detection. Models like Siamese networks, Siamese CNNs, and their variants have been developed to effectively capture common salient features across multiple images. Graph-based representations have gained popularity for modeling relationships between objects or regions in co-salient object detection. Adapting and innovating co-salient detection methods for diverse domains unlock new possibilities for applications such as video summarization, 3D object recognition, and medical diagnosis.

Graph convolutional networks (GCNs) and graph neural networks (GNNs) are employed to capture contextual information and propagate features across related images or regions. In re-thinking co-salient object detection, it's essential to consider not only advancements in model architectures and learning techniques but also the broader context of multimodal data, weak supervision, user interaction, scalability, and cross-domain applications.

TITLE : RETHINKING RGB-D SALIENT OBJECT DETECTION: MODELS, DATA SETS, AND LARGE-SCALE BENCHMARKS

AUTHOR: D.-P. Fan, Z. Lin, Z. Zhang, M. Zhu, and M.-M. Cheng

YEAR : 2021

DESCRIPTION: Rethinking RGB-D salient object detection involves exploring innovative models, leveraging diverse datasets, and establishing large-scale benchmarks to advance the state-of-the-art in this field. Develop deep learning architectures that effectively fuse RGB and depth modalities to exploit complementary information for salient object detection. Models like RGB-D CNNs, fusion-based CNNs, and attention mechanisms for cross-modal feature integration have shown promising results. Explore graph-based approaches, such as graph convolutional networks (GCNs) or graph neural networks (GNNs), to model relationships between pixels or regions in the RGB-D space. Graph-based representations enable capturing contextual information and facilitating information propagation for more accurate salient object detection.

Metrics should account for accuracy, robustness, consistency, and computational efficiency to provide comprehensive assessments of model performance. By rethinking RGB-D salient object detection through the lens of advanced models, diverse datasets, and comprehensive benchmarks, researchers can accelerate progress in this domain, leading to more accurate, robust, and practical solutions with broader applicability across various real-world scenarios.

TITLE : SUPPRESS AND BALANCE: A SIMPLE GATED NETWORK FOR SALIENT OBJECT DETECTION

AUTHOR: X. Zhao, Y. Pang, L. Zhang, H. Lu, and L. Zhang

YEAR : 2020

DESCRIPTION: The "Suppress and Balance: A Simple Gated Network for Salient Object Detection" proposes a straightforward yet effective approach for salient object detection using a gated network architecture. The proposed model architecture consists of a gated network, which is a variant of convolutional neural networks (CNNs) tailored for salient object detection. The gated network comprises multiple layers of convolutional and pooling operations, followed by a gating mechanism that selectively suppresses or enhances features based on their saliency relevance. The gating mechanism within the network is designed to dynamically suppress or enhance features at different spatial locations and feature channels. The gating mechanism operates based on learned gating weights, which are adaptively adjusted during the training process to focus on salient regions while suppressing background noise. One of the main highlights of the proposed approach is its simplicity and efficiency, making it computationally lightweight and easy to implement.

The proposed approach is evaluated on benchmark datasets for salient object detection, such as MSRA-B, ECSSD, and DUT-OMRON, to assess its performance against state-of-the-art methods. Performance metrics such as precision, recall, F-measure, and mean absolute error (MAE) are typically used to evaluate the effectiveness of the model in capturing salient objects accurately.

TITLE : LABEL DECOUPLING FRAMEWORK FOR SALIENT OBJECT DETECTION

AUTHOR : J. Wei, S. Wang, Z. Wu, C. Su, Q. Huang, and Q. Tian

YEAR : 2020

DESCRIPTION: The Label Decoupling Framework for Salient Object Detection introduces a novel approach to address the challenges of salient object detection by decoupling the learning process from the binary saliency labels. Traditional salient object detection methods rely on binary labels (salient vs. non-salient) for supervised learning, which may not fully capture the nuanced characteristics of salient objects. The Label Decoupling Framework aims to overcome this limitation by decoupling the learning process from binary labels, allowing the model to learn more informative and nuanced representations of salient objects. The framework typically employs a deep learning architecture, such as convolutional neural networks (CNNs), tailored for multi-label learning.

The performance of the Label Decoupling Framework is evaluated on standard benchmark datasets for salient object detection, such as MSRA-B, ECSSD, and DUT-OMRON. Performance metrics such as precision, recall, F-measure, and mean absolute error (MAE) are used to assess the effectiveness of the framework in capturing salient objects accurately across multiple saliency cues. The Label Decoupling Framework represents a novel approach to salient object detection, leveraging label decoupling and multi-label learning to capture the diverse and nuanced characteristics of salient objects more effectively compared to traditional binary label-based methods.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISITING SYSTEM

Although current salient object detection (SOD) works have achieved significant progress, they are limited when it comes to the integrity of the predicted salient regions. We define the concept of integrity at both a micro and macro level. Specifically, at the micro level, the model should highlight all parts that belong to a certain salient object. Meanwhile, at the macro level, the model needs to discover all salient objects in a given image. To facilitate integrity learning for SOD, we design a novel Integrity Cognition Network (ICON), which explores three important components for learning strong integrity features. 1) Unlike existing models, which focus more on feature discriminability, we introduce a diverse feature aggregation (DFA) component to aggregate features with various receptive fields (i.e., kernel shape and context) and increase feature diversity. Such diversity is the foundation for mining the integral salient objects. 2) Based on the DFA features, we introduce an integrity channel enhancement (ICE) component with the goal of enhancing feature channels that highlight the integral salient objects, while suppressing the other distracting ones. 3) After extracting the enhanced features, the part-whole verification (PWV) method is employed to determine whether the part and whole object features have strong agreement. Such part-whole agreements can further improve the micro-level integrity for each salient object. To demonstrate the effectiveness of our ICON, comprehensive experiments are conducted on seven challenging benchmarks. Our ICON outperforms the baseline methods in terms of a wide range of metrics.

3.1.1. DISADVANTAGES OF EXISTING SYSTEM

- They apply generalized approach for this problems.
- They did not deploy the model.
- Their process require lot of compute power.
- Limited scalability.

3.2 PROPOSED SYSTEM

The proposed system aims to employ advanced neural network techniques for the classification of Nazi infantry and artillery weapons. Leveraging deep learning architectures, the system will analyze historical images, schematics, and textual descriptions to accurately categorize and label these weapons. The neural network will be trained on a comprehensive dataset comprising diverse examples of Nazi weaponry. Through convolutional neural networks (CNNs) for image analysis and recurrent neural networks (RNNs) for text processing, the system will learn intricate features and relationships within the data.

This hybrid approach will enable precise classification, distinguishing between various types of rifles, handguns, machine guns, and artillery pieces. By harnessing the power of neural networks, the proposed system seeks to enhance our understanding of these historical artifacts while contributing to the preservation of historical knowledge.

3.2.1. ADVANTAGES

- To build a framework based application for deployment purposes.
- It focused on only Artillery weapons classification.

- High scalability.
- It compares more than a two architecture to getting better accuracy level.

3.3 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out.

3.3.1 Splitting the dataset:

The data use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. It has the test dataset (or subset) in order to test our models and it will do this using Tensor flow library in Python using the Keras method.

This set is used to tune hyperparameters and monitor the performance of the model during training. It helps prevent overfitting by providing an independent dataset not used in training. Usually, around 10-15% of the data is allocated to the validation set. This set is used to evaluate the final performance of the trained model.

3.3.2 Construction of a Detecting Model:

Deep learning needs data gathering have lot of past image data's. Training and testing this model working and predicting correctly. Clearly define the detection problem you want to solve. Gather relevant data for training your detection model. This may involve collecting labeled data if it's available or creating annotations yourself. Preprocess the data to make it suitable for training. Choose an appropriate model architecture for your detection task.

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

Design is meaningful engineering representation of something that is to be built. Software design is a process design is the perfect way to accurately translate requirements in to a finished software product. Design creates a representation or model, provides detail about software data structure, architecture, interfaces and components that are necessary to implement a system.

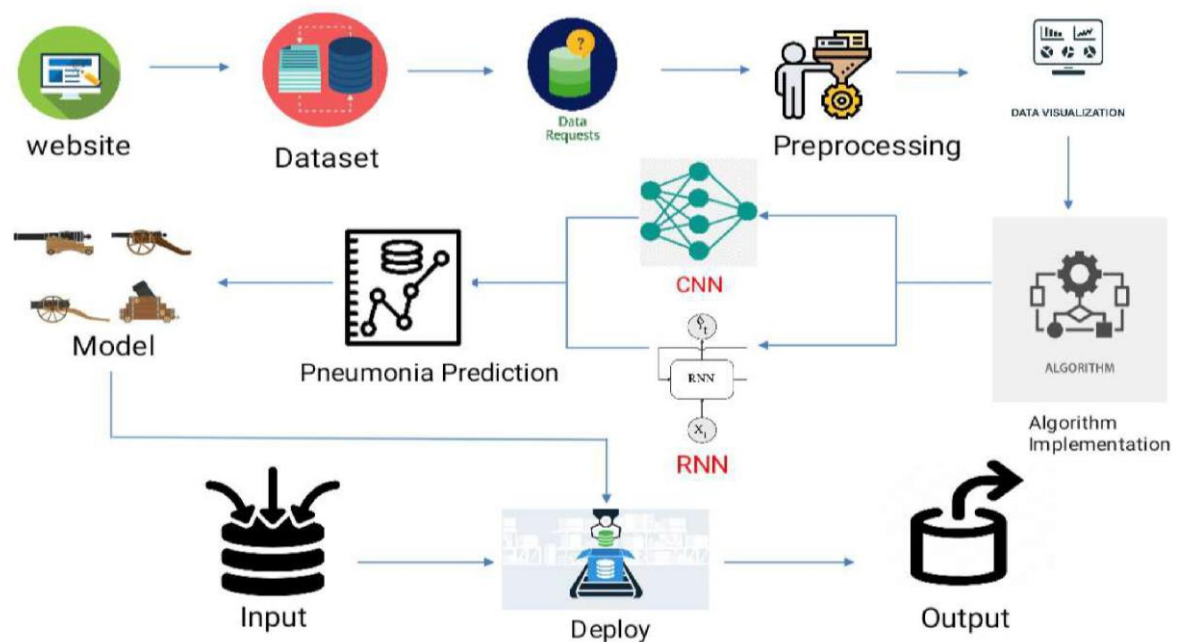


Fig.4.1 System Architecture

4.2 UML DIAGRAMS

4.2.1 DATAFLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model. Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top down approach to Systems Design.

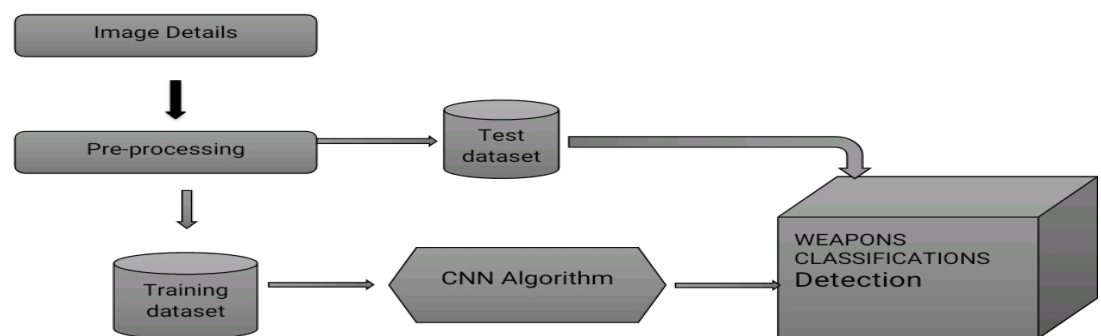


Fig: Process of dataflow diagram

Fig.4.2 DATAFLOW DIAGRAM

4.2.2 WORKFLOW DIAGRAM

A workflow diagram is a visual representation of a process or system that illustrates the sequence of steps or activities required to complete a particular task or achieve a specific outcome. It provides a structured way of understanding how different elements in a process interact and the order in which they occur.

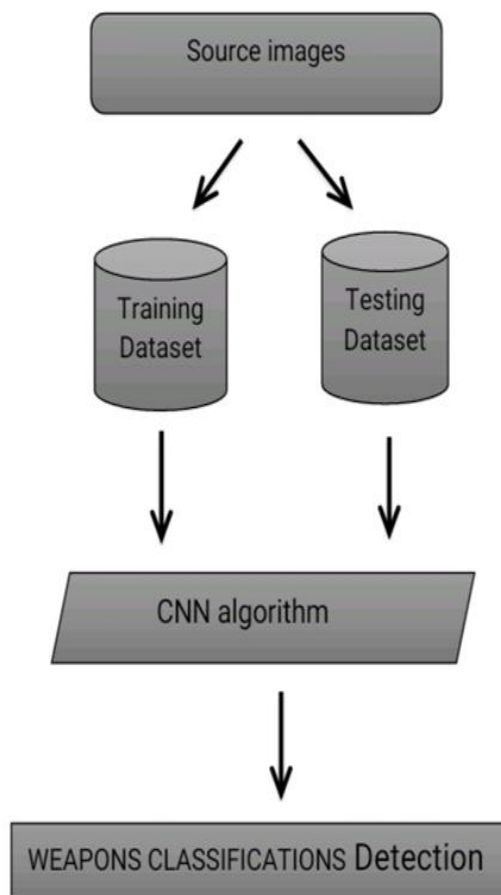


Fig.4.3 WORKFLOW DIAGRAM

4.2.3 USE CASE DIAGRAM

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

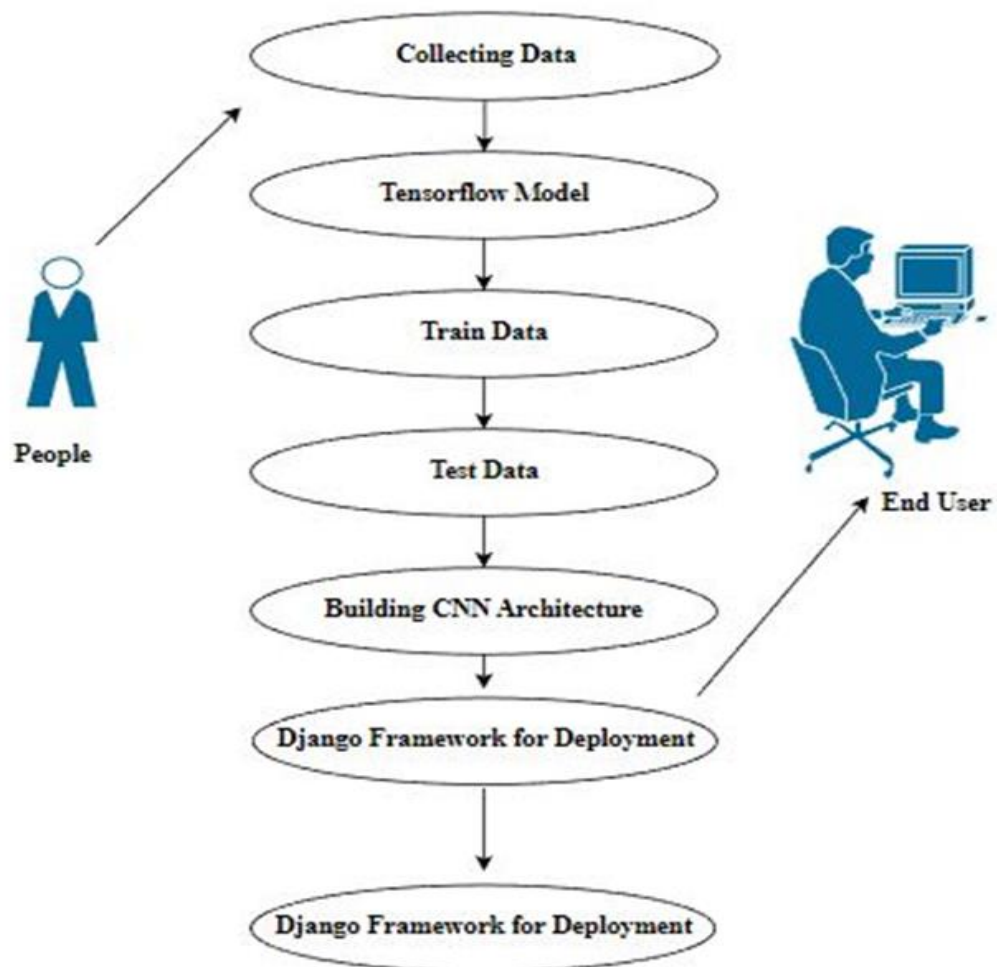


Fig.4.4 USE CASE DIAGRAM

4.2.4 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.

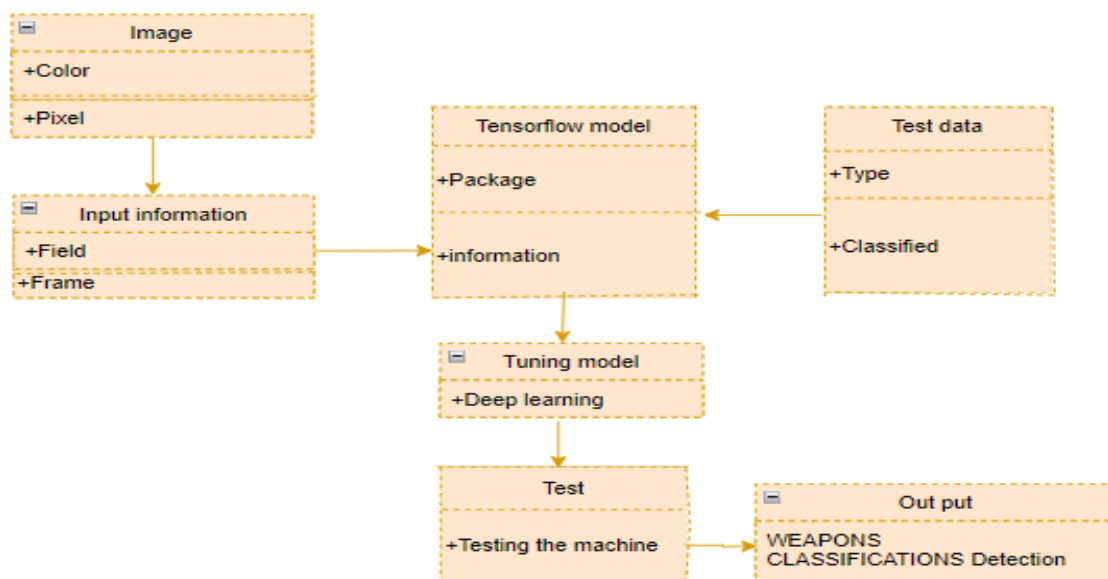


Fig 4.5 CLASS DIAGRAM

4.2.5 ACTIVITY DIAGRAM

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

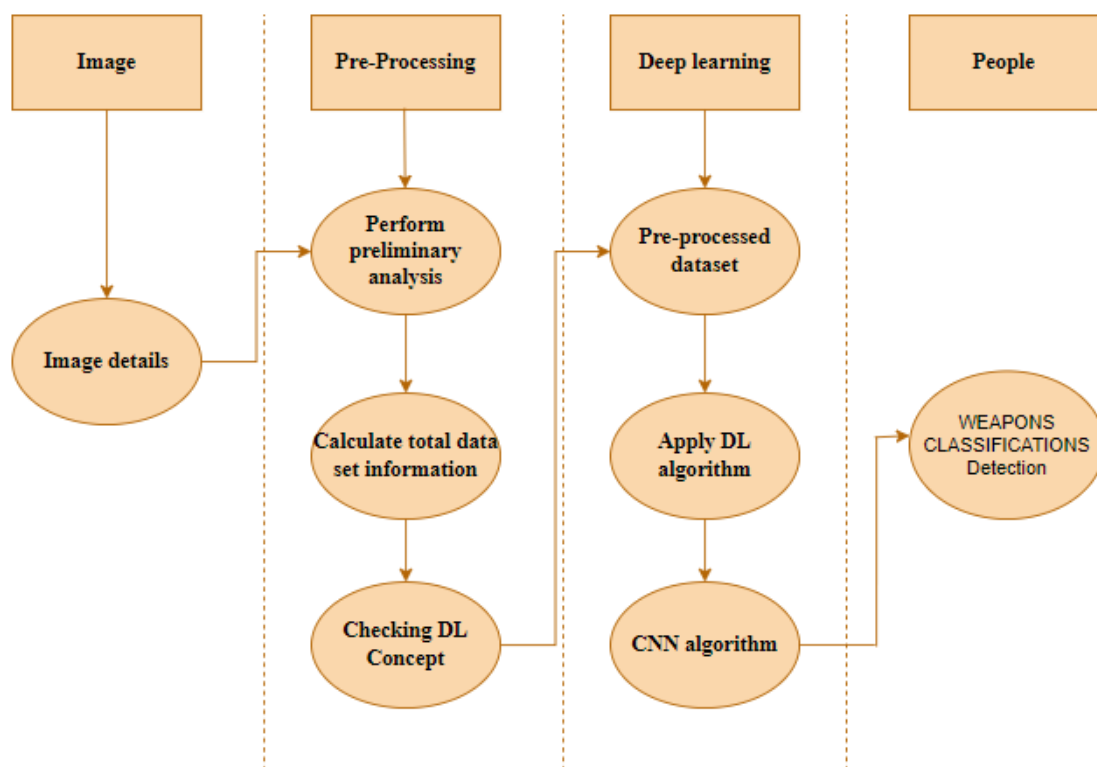


Fig 4.6 ACTIVITY DIAGRAM

4.2.6 SEQUENCE DIAGRAM

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

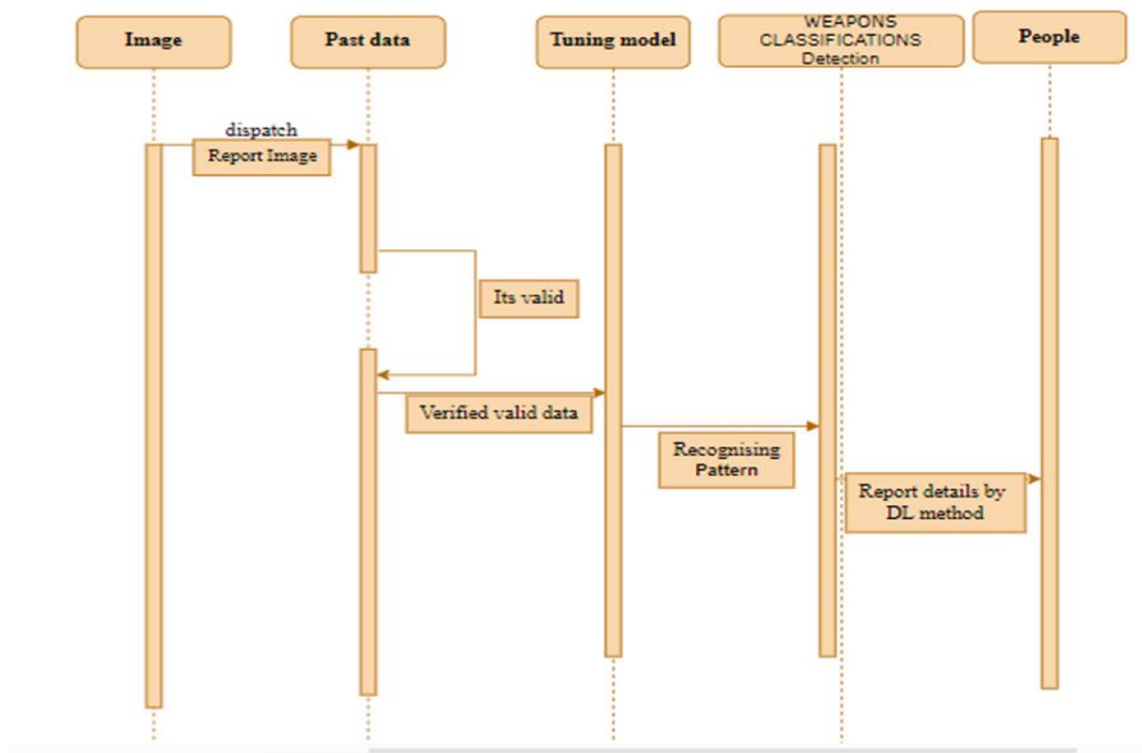


Fig 4.7 SEQUENCE DIAGRAM

4.2.7 ER DIAGRAM

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

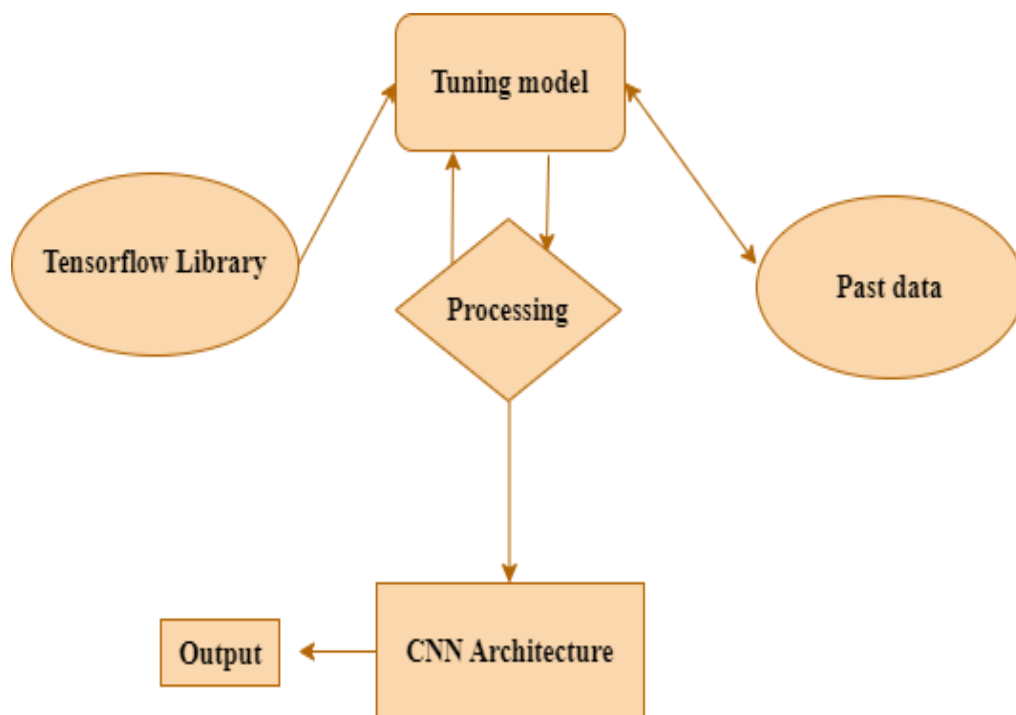


Fig 4.8 ER DIAGRAM

4.2.8 COLLABORATION DIAGRAM

A collaboration diagram show the objects and relationships involved in an interaction, and the sequence of messages exchanged among the objects during the interaction.

The collaboration diagram can be a decomposition of a class, class diagram, or part of a class diagram.it can be the decomposition of a use case, use case diagram, or part of a use case diagram.

The collaboration diagram shows messages being sent between classes and object (instances). A diagram is created for each system operation that relates to the current development cycle (iteration).

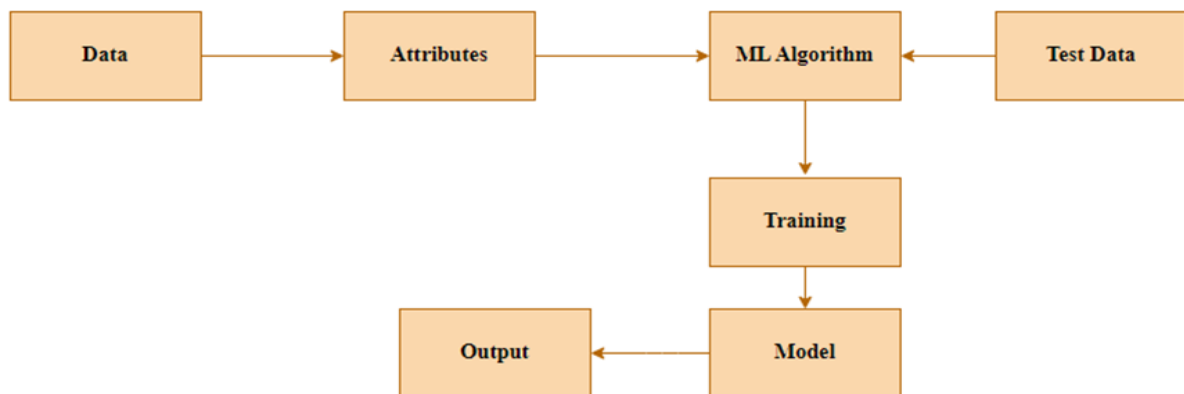


Fig 4.9 COLLABORATION DIAGRAM

CHAPTER 5

SYSTEM REQUIREMENTS

5.1 HARDWARE REQUIREMENTS

The most common set of requirements is defined by the any system or the software application is the physical computer resources, also known as hardware, a hardware requirements list is often accompanied by a Hardware Compatibility List (HCL), especially in case of operating system.

| | |
|-----------|------------------|
| Processor | : Pentium IV/III |
| Hard disk | : minimum 80 GB |
| RAM | : minimum 2 GB |

5.2 SOFTWARE REQUIREMENTS

The software requirements deals with defining software resource requirements and prerequisites that needs to be installed on a computer to provide optimal functioning of an application.

| | |
|------------------|----------------------------------|
| Operating System | : Windows |
| Tool | : Anaconda with Jupyter Notebook |

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 SOFTWARE DESCRIPTION

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system “Conda”. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the `conda build` command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

6.1.1 ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line

commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- Spyder
- PyCharm
- VSCode
- Glueviz

- Orange 3 App
- RStudio
- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution.

Navigator allows you to launch common Python programs and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository.

Anaconda comes with many built-in packages that you can easily find with `conda list` on your anaconda prompt. As it has lots of packages (many of which are rarely used), it requires lots of space and time as well. If you have enough space, time and do not want to burden yourself to install small utilities like JSON, YAML, you better go for Anaconda.

6.1.2 JUPYTER NOTEBOOK

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results

(figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

Installation: The easiest way to install the Jupyter Notebook App is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called **Anaconda**

Running the Jupyter Notebook

Launching Jupyter Notebook App: The Jupyter Notebook App can be launched by clicking on the Jupyter Notebook icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (*cmd* on Windows): “jupyter notebook”. This will launch a new browser window (or a new tab) showing the Notebook Dashboard, a sort of control panel that allows (among other things) to select which notebook to open.

When started, the Jupyter Notebook App can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders.

Otherwise, you need to choose a Jupyter Notebook App start-up folder which will contain all the notebooks.

Save notebooks: Modifications to the notebooks are automatically saved every few minutes.

To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy...) and save the modifications on the copy.

Executing a notebook: Download the notebook you want to execute and put it in your notebook folder (or a sub-folder of it).

Purpose: To support interactive data science and scientific computing across all programming languages.

File Extension: An IPYNB file is a notebook document created by Jupyter Notebook, an interactive computational environment that helps scientists manipulate and analyze data using Python.

JUPYTER Notebook App: The *Jupyter Notebook App* is a server-client application that allows editing and running notebook documents via a web browser. The *Jupyter Notebook App* can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the *Jupyter Notebook App* has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

kernel: A notebook kernel is a “computational engine” that executes the code contained in a Notebook document. The ipython kernel, referenced in this guide, executes python code. Kernels for many other languages exist ([official kernels](#)). When you open a Notebook document, the associated *kernel* is automatically launched.

When the notebook is *executed* (either cell-by-cell or with menu *Cell -> Run All*), the *kernel* performs the computation and produces the results. Depending on the type of computations, the *kernel* may consume significant CPU and RAM. Note that the RAM is not released until the *kernel* is shut-down.

Notebook Dashboard: The Notebook Dashboard is the component which is shown first when you launch Jupyter Notebook App. The Notebook Dashboard is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown). The Notebook Dashboard has other features similar to a file manager, namely navigating folders and renaming/deleting files

Working Process:

- Download and install anaconda and get the most useful package for machine learning in Python.
- Load a dataset and understand its structure using statistical summaries and data visualization.
- Machine learning models, pick the best and build confidence that the accuracy is reliable.

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems. There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

The best way to get started using Python for machine learning is to complete a project.

- It will force you to install and start the Python interpreter (at the very least).
- It will give you a bird's eye view of how to step through a small project.
- It will give you confidence, maybe to go on to your own small projects.

When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

- Define Problem.
- Prepare Data.
- Evaluate Algorithms.
- Improve Results.
- Present Results.

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

Here is an overview of what we are going to cover:

1. Installing the Python anaconda platform.
2. Loading the dataset.
3. Summarizing the dataset.
4. Visualizing the dataset.
5. Evaluating some algorithms.
6. Making some predictions.

PYTHON

Introduction:

Python is an interpreted high-level general-purpose programming language.

Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is

not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020. Python consistently ranks as one of the most popular programming languages

Design Philosophy & Feature

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta-programming and meta-objects (magic methods)).

Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible (with modules). This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications.

Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" design philosophy.

Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture. "Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the C-Python reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

Python's developers aim to keep the language fun to use. This is reflected in its name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (a reference to a Monty Python sketch) instead of the standard foo and bar.

A common neologism in the Python community is *pythonic*, which can have a wide range of meanings related to program style. To say that code is *pythonic* is to say that it uses Python idioms well, that it is natural or shows fluency in the

language, that it conforms with Python's minimalist philosophy and emphasis on readability.

In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic.

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as Pythonistas. Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will.^{[80][81]} However, better support for co-routine-like functionality is provided, by extending Python's generators.

Before 2.5, generators were lazy iterators; information was passed unidirectionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

Python has array index and array slicing expressions on lists, denoted as `a[Key]`, `a[start:stop]` or `a[start:stop:step]`. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the start index up to, but not including, the stop index.

The third slice parameter, called step or stride, allows elements to be skipped and reversed. Slice indexes may be omitted, for example `a[:]` returns a copy of the entire list. Each element of a slice is a shallow copy.

6.2 LIST OF MODULES

1. Manual Net
2. GOOGLE NET
3. MOBILE NET
4. Deploy

6.3 MODULE DESCRIPTION

6.3.1. 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 adding layers of CNN



Fig 6.1 IMPORT THE GIVEN IMAGE FROM DATASET

6.3.2. TO TRAIN THE MODULE BY GIVEN IMAGE DATASET:

To train our dataset using classifier and fit generator function also we make training steps per epoch's then total number of epochs, validation data and validation steps using this data we can train our dataset.



Fig 6.2 TO TRAIN THE MODULE BY GIVEN IMAGE DATASET

6.3.3. WORKING PROCESS OF LAYERS IN CNN MODEL:

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human severity and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. Their network consists of four layers with 1,024 input units, 256 units in the first hidden layer, eight units in the second hidden layer, and two output units.

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 $28 \times 28 = 784$, it need to convert it into 784×1 before feeding into input.

Convo Layer:

Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive field (it is a local region of the input image that has the same size as that of filter) and the filter. Result of the operation is single integer of the output volume.

Then the filter over the next receptive field of the same input image by a Stride and do the same operation again. It will repeat the same process again and again until it goes through the whole image. The output will be the input for the next layer.

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 reducing to 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.

Output Layer:

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

6.3.4. PNEUMONIA CLASSIFICATION IDENTIFICATION:

We give input image using keras pre-processing package. That input Image converted into array value using pillow and image to array function package. We have already classified image dataset. It classifies what are the weapons classification Detection.

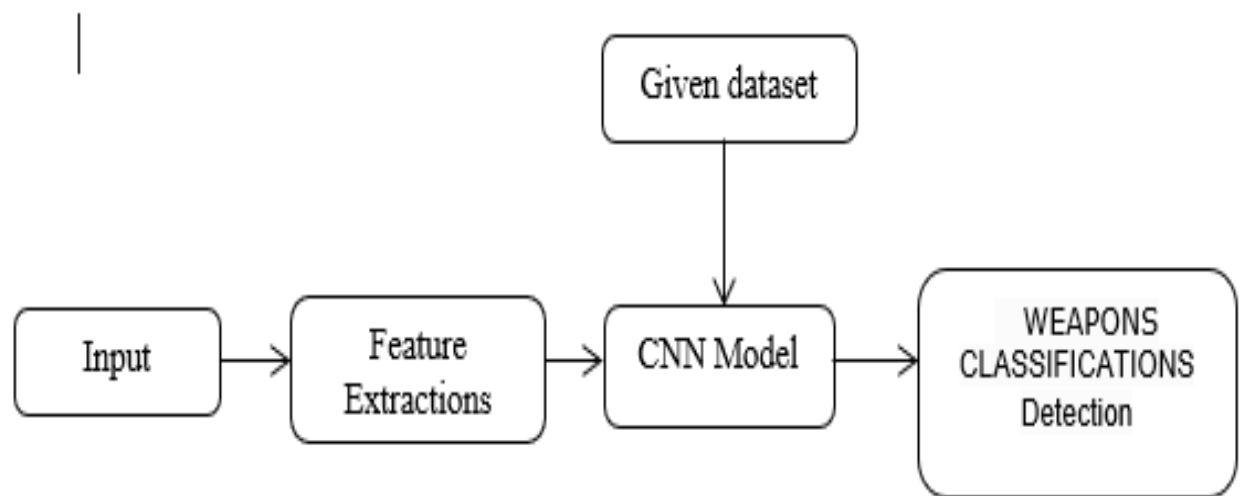


Fig: 6.3 PNEUMONIA CLASSIFICATION IDENTIFICATION

The weapons classification recognition method is based on a two-channel architecture that is able to recognize classification of weapons. The PNEUMONIA images are used as the input into the inception layer of the CNN. The Training phase involves the feature extraction and classification using convolution neural network.

CHAPTER 7

DEPLOYMENT

7.1 Django(WEB FRAMEWORK)

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

FEATURES:

Complete

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up-to-date documentation.

Versatile

Django can be (and has been) used to build almost any type of website from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc). The site you are currently reading is built with Django. Internally, while it provides choices for almost any functionality you might want (e.g. several popular databases, templating engines, etc.), it can also be extended to use other components if needed.

Secure

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage

user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash. A password hash is a fixed-length value created by sending the password through a cryptographic hash function. Django can check if an entered password is correct by running it through the hash function and comparing the output to the stored hash value. However due to the "one-way" nature of the function, even if a stored hash value is compromised it is hard for an attacker to work out the original password. Django enables protection against many vulnerabilities by default, including SQL injection, cross-site scripting, cross-site request forgery and clickjacking (see Website security for more details of such attacks).

Scalable

Django uses a component-based “shared-nothing” architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers. Some of the busiest sites have successfully scaled Django to meet their demands (e.g. Instagram and Disqus, to name just two).

Maintainable

Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern).

Portable

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavours of Linux, Windows, and Mac OS X. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

7.2 WORKING PROCESS

- Download and install anaconda and get the most useful package for machine learning in Python.
- Load a dataset and understand its structure using statistical summaries and data visualization.
- Machine learning models, pick the best and build confidence that the accuracy is reliable.

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems. There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

Here is an overview of what we are going to cover:

1. Installing the Python anaconda platform.
2. Loading the dataset.
3. Summarizing the dataset.
4. Visualizing the dataset.
5. Evaluating some algorithms.
6. Making some predictions.

CHAPTER 8

CODING AND RESULT

APPENDICES

A.SAMPLE

MODULE 1:

MANUAL NET ARCHITECTURE

```
import warnings

warnings.filterwarnings('ignore')

import os

import glob

import numpy as np

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from PIL import Image

from tensorflow.keras.layers import Convolution2D

from tensorflow.keras.layers import MaxPooling2D

from tensorflow.keras.layers import Flatten

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Activation

from keras.callbacks import ModelCheckpoint

import matplotlib.pyplot as plt

Abwehrflammenwerfer_42 = 'DATASET/TRAIN/Abwehrflammenwerfer_42'

AK_47 = 'DATASET/TRAIN/AK-47'

Astra_900 = 'DATASET/TRAIN/Astra_900'

Barnitzke = 'DATASET/TRAIN/Barnitzke'

Breda_M37 = 'DATASET/TRAIN/Breda_M37'

Beretta_Model_38 = 'DATASET/TRAIN/Beretta_Model_38'

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(3, n, idx+1)

    img = plt.imread(img_path)

    plt.imshow(img, cmap='gray')

    plt.axis('off')

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
    data['Max_width']=0
    data['Min_height']=10**100
    data['Max_height']=0

    for f in files:
        img=Image.open(f)
        width,height=img.size
        data['Min_width']=min(width,data['Min_width'])
        data['Max_width']=max(width, data['Max_width'])
        data['Min_height']=min(height, data['Min_height'])
        data['Max_height']=max(height, data['Max_height'])
    image_details_print(data,path)
    print("")
    print("TRAINING DATA FOR Abwehrflammenwerfer 42:")
    print("")
    images_details(Abwehrflammenwerfer_42)
    print("")
    plot_images(Abwehrflammenwerfer_42, 10)
    print("")
    print("TRAINING DATA FOR AK-47:")
    print("")
    images_details(AK_47)
    print("")
    plot_images(AK_47, 10)
    print("")
    print("TRAINING DATA FOR Astra_900:")
    print("")
    images_details(Astra_900)
    print("")
    plot_images(Astra_900, 10)
    print("")

```

```

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

```

MODULE 2:

GOOGLE OR INCEPTION NET ARCHITECTURE

```

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 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')

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 = 'DATASET/TRAIN',
target_size=(224,224),
    batch_size=32, class_mode='categorical')

```

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

```
def googlenet(input_shape, n_classes):
```

```
    def inception_block(x, f):
```

```
        t1 = Conv2D(f[0], 1, activation='relu')(x)
```

```
        t2 = Conv2D(f[1], 1, activation='relu')(x)
```

```
        t2 = Conv2D(f[2], 3, padding='same', activation='relu')(t2)
```

```
        t3 = Conv2D(f[3], 1, activation='relu')(x)
```

```
        t3 = Conv2D(f[4], 5, padding='same', activation='relu')(t3)
```

```
        t4 = MaxPool2D(3, 1, padding='same')(x)
```

```
        t4 = Conv2D(f[5], 1, activation='relu')(t4)
```

```
    output = Concatenate()([t1, t2, t3, t4])
```

```
    return output
```

```
input = Input(input_shape)
```

```
x = Conv2D(64, 7, strides=2, padding='same', activation='relu')(input)
```

```
x = MaxPool2D(3, strides=2, padding='same')(x)
```

```
x = Conv2D(64, 1, activation='relu')(x)
```

```
x = Conv2D(192, 3, padding='same', activation='relu')(x)
```

```
x = MaxPool2D(3, strides=2)(x)
```

```
x = inception_block(x, [64, 96, 128, 16, 32, 32])
```

```
x = inception_block(x, [128, 128, 192, 32, 96, 64])
```

```
x = MaxPool2D(3, strides=2, padding='same')(x)
```

```
x = inception_block(x, [192, 96, 208, 16, 48, 64])
```

```
x = inception_block(x, [160, 112, 224, 24, 64, 64])
```

```
x = inception_block(x, [128, 128, 256, 24, 64, 64])
```

```
x = inception_block(x, [112, 144, 288, 32, 64, 64])
```

```
x = inception_block(x, [256, 160, 320, 32, 128, 128])
```

```
x = MaxPool2D(3, strides=2, padding='same')(x)
```

```
x = inception_block(x, [256, 160, 320, 32, 128, 128])
```

```
x = inception_block(x, [384, 192, 384, 48, 128, 128])
```

```
x = AvgPool2D(7, strides=1)(x)
```

```
x = Dropout(0.4)(x)
```



```

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

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

```

MODULE 3

MOBILENET ARCHITECTURE

```

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 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')

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 = 'DATASET/TRAIN',target_size=(224,224),
                                     batch_size=32,class_mode='categorical')
```

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

```
def mobilenet(input_shape, n_classes):
```

```
    def mobilenet_block(x, f, s=1):
```

```
        x = DepthwiseConv2D(3, strides=s, padding='same')(x)
```

```
        x = BatchNormalization()(x)
```

```
        x = ReLU()(x)
```

```
        x = Conv2D(f, 1, strides=1, padding='same')(x)
```

```
        x = BatchNormalization()(x)
```

```
        x = ReLU()(x)
```

```
    return x
```

```
input = Input(input_shape)
```

```
x = Conv2D(32, 3, strides=2, padding='same')(input)
```

```
x = BatchNormalization()(x)
```

```
x = ReLU()(x)
```

```
x = mobilenet_block(x, 64)
```

```
x = mobilenet_block(x, 128, 2)
```

```
x = mobilenet_block(x, 128)
```

```
x = mobilenet_block(x, 256, 2)
```

```
x = mobilenet_block(x, 256)
```


```
x = mobilenet_block(x, 512, 2)
```

```
for _ in range(5):
```

```
    x = mobilenet_block(x, 512)
```

B. SCREENSHOTS

TO ANALYSE NAZI INFANTRY AND ARTILLERY WEAPONS CLASSIFICATIONS




DESCRIPTION

Nazi Germany during World War II deployed a wide array of infantry and artillery weaponry. Infantry rifles like the Karabiner 98k and the innovative StG 44 assault rifle served as standard issue firearms, while submachine guns such as the MP40 and MP38 offered close-quarters firepower. Machine guns like the MG34 and the MG42 provided sustained automatic fire support. Pistols like the Luger P08 and Walther P38 were officer sidearms, complemented by various hand grenades. The Nazi arsenal also included formidable tanks like the Tiger I and Panther, anti-tank rocket launchers, heavy artillery like the 88mm Flak gun, mortars, anti-tank guns, flamethrowers, and other specialized weaponry, all contributing to their formidable military machine during the war.

EXPLORE

EXAMPLES OF WEAPONS CLASSIFICATIONS



WEAPONS CLASSIFICATIONS

CONTRIBUTORS DOMAIN_RESULT PROBLEM_STATEMENTS LOGOUT

MILITARY WEAPONS CLASSIFICATIONS

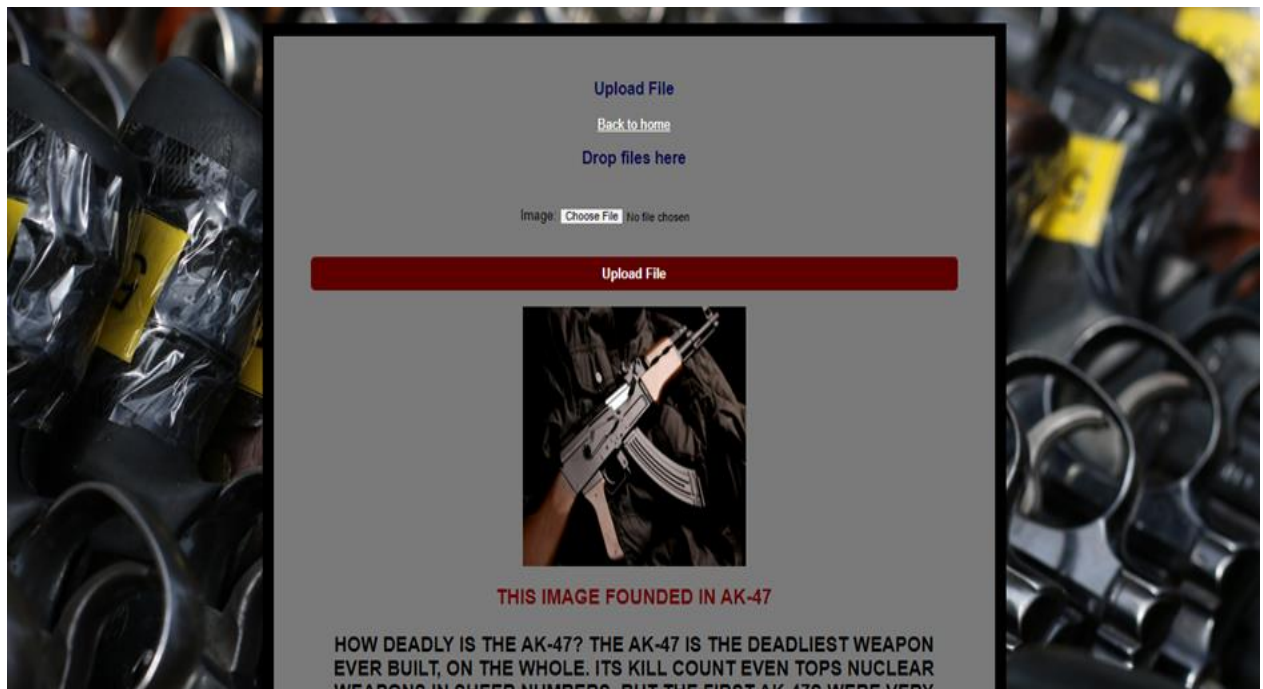
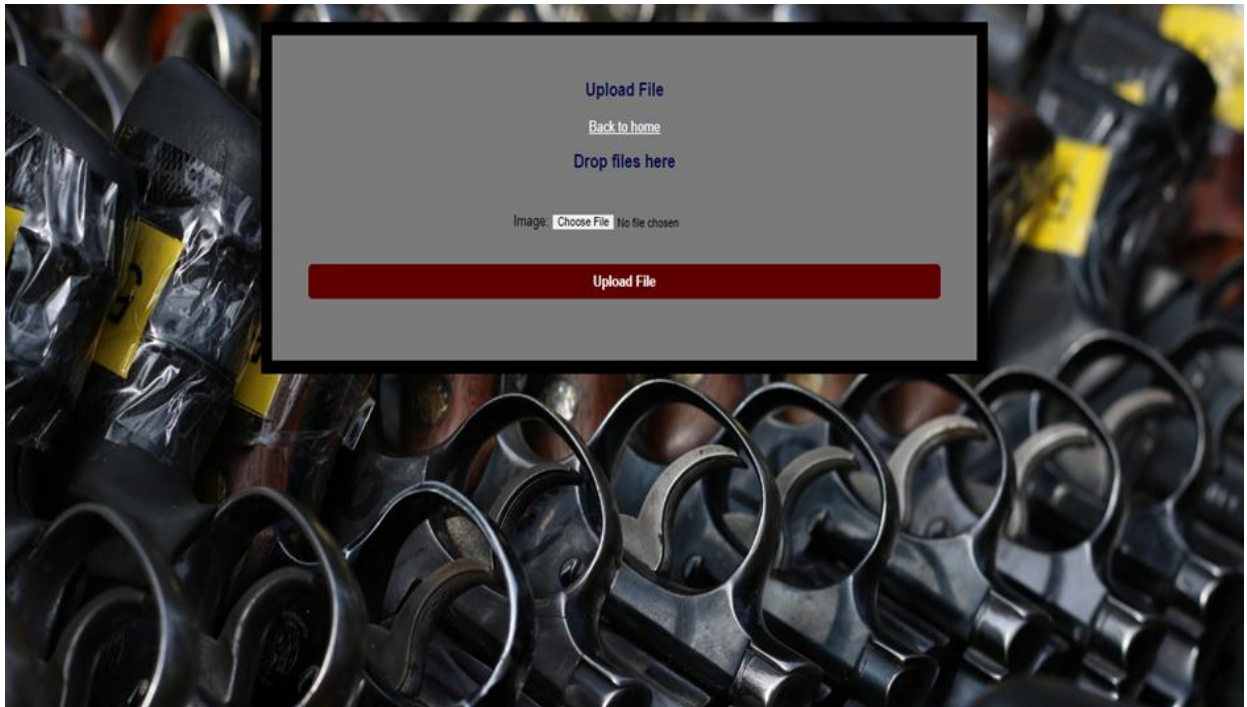
Weapons are instruments designed to harm, kill, or otherwise disable other human beings, to destroy other military resources, or to deter an enemy's ability to make war through the actual or threatened destruction of crucial components of their society.

DEPLOYMENT

Machine learning deployment is the act of making a trained machine learning model available for use in real-world applications or systems to make predictions or automate tasks.

DATABASE

A database is a structured collection of data organized for efficient storage, retrieval, and management of information.



CHAPTER 9

CONCLUSION AND FUTURE ENHANCEMENT

CONCLUSION:

The application of neural network techniques for the classification of Nazi infantry and artillery weapons represents a significant advancement in historical research and technological analysis. By leveraging machine learning algorithms, we aim to create a systematic and efficient method for identifying and categorizing these historical artifacts, contributing to the preservation of knowledge about World War II weaponry. This approach not only streamlines the process of cataloging but also provides a valuable tool for researchers, historians, and enthusiasts to gain insights into the technological aspects of Nazi weaponry.

It is essential to emphasize the responsible and ethical use of such technology, ensuring that the study focuses on historical understanding and education rather than glorification or promotion of ideologies associated with the Nazi regime. The project's scope extends beyond mere technological innovation, encompassing historical preservation, educational applications, and the promotion of sensitivity when dealing with artifacts linked to contentious historical periods.

In essence, the integration of neural network techniques into the classification of Nazi infantry and artillery weapons enhances our ability to explore and comprehend the complexities of military history, contributing to a more nuanced understanding of the technological landscape during a critical period in human history.

FUTURE ENHANCEMENT:

Future work in the classification of Nazi infantry and artillery weapons using neural network techniques should prioritize dataset expansion to encompass a broader array of weapon variations, fine-tune the model for improved accuracy, and explore multimodal analysis by integrating textual descriptions and contextual data. The development of a user-friendly interface for researchers and

enthusiasts, collaboration with institutions for data sharing, and a focus on ethical considerations and educational components are essential elements for enhancing the project's impact. Additionally, extending the neural network application to classify weapons from different historical periods and implementing robust validation procedures with expert input will contribute to a more sophisticated and reliable tool for historical research.

- To apply this method in cloud computing
- To apply this method in IOT system

References:

- [1] H. U. Khan, M. Z. Malik, S. Nazir and F. Khan, "Utilizing Bio Metric System for Enhancing Cyber Security in Banking Sector: A Systematic Analysis," in IEEE Access, vol. 11, pp. 80181-80198, 2024, doi: 10.1109/ACCESS.2023.3298824.
- [2] N. A. Karim, O. A. Khashan, H. Kanaker, W. K. Abdulraheem, M. Alshinwan and A. -K. Al-Banna, "Online Banking User Authentication Methods: A Systematic Literature Review," in IEEE Access, vol. 12, pp. 741-757, 2024, doi: 10.1109/ACCESS.2023.3346045.
- [3] Z. Wang, M. Muhammad, N. Yadikar, A. Aysa and K. Ubul, "Advances in Offline Handwritten Signature Recognition Research: A Review," in IEEE Access, vol. 11, pp. 120222-120236, 2023, doi: 10.1109/ACCESS.2023.3326471.
- [4] A. Almadan and A. Rattani, "Benchmarking Neural Network Compression Techniques for Ocular-Based User Authentication on Smartphones," in IEEE Access, vol. 11, pp. 36550-36565, 2023, doi:10.1109/ACCESS.2023.3265357.
- [5] P. Ody, F. Gorski and A. Czyżewski, "User Authentication by Eye Movement Features Employing SVM and XGBoost Classifiers," in IEEE Access, vol. 11, pp. 93341-93353, 2023, doi: 10.1109/ACCESS.2023.3309000.
- [6] W. Mu and B. Liu, "Voice Activity Detection Optimized by Adaptive Attention Span Transformer," in IEEE Access, vol. 11, pp. 31238-31243, 2023, doi: 10.1109/ACCESS.2023.3262518.
- [7] C. Busch and B. Liu, "Design of a Batteryless, Wireless, and Secure System-on-Chip Implant for In-Body Strain Sensing," 2023 Working Conference on Software Visualization (VISOFT), Luxembourg, 2023, pp. 125-129, doi: 10.1109/VISOFT52517.2023.00024.
- [8] D. Benalcazar, J. E. Tapia, S. Gonzalez and C. Busch, "Synthetic ID Card Image Generation for Improving Presentation Attack Detection," in IEEE Transactions on Information Forensics and Security, vol. 18, pp. 1814-1824, 2023, doi: 10.1109/TIFS.2023.3255585.
- [9] K. Malinka, O. Hujnak, P. Hanacek and L. Hellebrandt, "E-Banking Security Study—10 Years Later," in IEEE Access, vol. 10, pp. 16681-16699, 2022, doi: 10.1109/ACCESS.2022.3149475.

- [10] A. Sedik et al., "Deep Learning Modalities for Biometric Alteration Detection in 5G Networks-Based Secure Smart Cities," in *IEEE Access*, vol. 9, pp. 94780-94788, 2021, doi: 10.1109/ACCESS.2021.3088341.
- [11] C. -W. Hung, J. -R. Wu and C. -H. Lee, "Device Light Fingerprints Identification Using MCU-Based Deep Learning Approach," in *IEEE Access*, vol. 9, pp. 168134-168140, 2021, doi: 10.1109/ACCESS.2021.3135448.