**<< PROJECT TITLE>>**

## A RESEARCH PROJECT REPORT

***Submitted by***

### K. AKSHITHA L. AKHILA R. RAVALI 21RH1A6626 21RH1A6632 21RH1A6651

#### Under the Esteemed Guidance of

#### Internal Guide Name

#### Designation

**In partial fulfilment of the academic requirements for the Degree of**

**BACHELOR OF TECHNOLOGY**

# CSE-AI & ML

****

**MALLA REDDY ENGINEERING COLLEGE FOR WOMEN**

**(Autonomous Institution-UGC, Govt. of India**)

##### Accredited by NAAC with ‘A+’ Grade

National Ranking by NIRF Innovation – Rank band (151-300), MHRD, Govt. of India

**Approved by AICTE, Permanently Affiliated to JNTUH, ISO 9001:2015 Certified Institution Maisammaguda, Dhulapally, Secunderabad, Kompally-500100**[**.**](http://www.mallareddyecw.com/)

## 2025-26

##### Accredited by NAAC with ‘A+’ Grade

National Ranking by NIRF Innovation – Rank band (151-300), MHRD, Govt. of India

**Approved by AICTE, Permanently Affiliated to JNTUH, ISO 9001:2015 Certified Institution Maisammaguda, Dhulapally, Secunderabad, Kompally-500100.**

**DEPARTMENT OF CSE - AI & ML**

##### CERTIFICATE

This is to certify that the Research project work entitled **<< PROJECT TITLE>>** is carried out by<< STUDENT NAME> (<H.T NOS>), L.AKHILA (21RH1A6632), R. RAVALI (21RH1A6651)

in partial fulfillment for the award of degree of BACHELOR OF TECHNOLOGY in CSE (AI & ML), Jawaharlal Nehru Technological University, Hyderabad during the academic year 2025- 2026.

|  |  |
| --- | --- |
| **Supervisor’s Signature Internal Guide Name**  **Designation** | **Head of the Department Dr. M,NARENDHAR AssociateProfessor** |
| **External Examiner** | |

##### Accredited by NAAC with ‘A+’ Grade

National Ranking by NIRF Innovation – Rank band (151-300), MHRD, Govt. of India

**Approved by AICTE, Permanently Affiliated to JNTUH, ISO 9001:2015 Certified Institution Maisammaguda, Dhulapally, Secunderabad, Kompally-500100.**

**Department of CSE –AI & ML DECLARATION**

We hereby declare that Research project entitled **<< PROJECT TITLE>>** submitted to Malla Reddy Engineering College For Women affiliated to Jawaharlal Nehru Technological University, Hyderabad (JNTUH) for the award of the Degree of Bachelor of Technology in CSE(AI & ML) is a result of original research work done by us. It is further declaredthat the Research Project report or any part thereof has not been previously submitted to any University or Institute for the award of Degree.

Student Name (H.T No)

Student Name (H.T No)

Student Name (H.T No)

**ACKNOWLEDGEMENT**

We feel ourselves honored and privileged to place our warm salutation to our college **Malla Reddy Engineering College for Women** and **Department of CSE-AI&ML**. which gave us the opportunity to have expertise in engineering and profound technical knowledge.

We would like to deeply thank our honorable MLA, **Sri.Ch. Malla Reddy Garu**, founder chairman MRGI, the largest cluster of institutions in the state of Telangana for providing us with all the resources in the college to make our project Success.

We wish to convey gratitude to our **Principal Dr. Y. Madhavee Latha**, for providing us with the environment and mean to enrich our skills and motivating us in our endeavor and helping us to realize our full potential.

We would like to thank **Prof. A. Radha Rani**, Director of Computer Science and Engineering & Information Technology for encouraging us to take up a project on this subject and motivating us towards the Project Work.

We express our sincere gratitude to **Dr. M.NARENDHAR**, Head of the Department of CSE- AI&ML for inspiring us to take up a project on this subject and successfully guiding us towards its completion,

We would like to thank our guide **Internal Guide Name**, Associate Professor and all the Faculty members for their valuable guidance and encouragement towards the completion of our project work.

**With Regards and Gratitude**

**STUDENT NAME(H.T NO) STUDENT NAME(H.T NO)**

**STUDENT NAME(H.T NO)**

## ABSTRACT

The increasing demand for effective triage in emergency departments has led to a growing interest in leveraging machine learning for patient classification. This project presents a robust machine learning pipeline for predicting Korean Triage and Acuity Scale (KTAS) levels based on both structured clinical features and unstructured textual data. The system preprocesses patient data using TF-IDF and sentence embeddings, applies advanced class balancing techniques, and evaluates multiple models including LightGBM, XGBoost, SVM, Logistic Regression, and Random Forest. The LightGBM model achieved the best performance with an accuracy of 83.97%, a weighted F1 score of 83.83%, and a Cohen’s Kappa score of 0.79. Evaluation metrics, confusion matrices, and calibration plots were generated for all models, enabling a comprehensive comparison. The final pipeline is integrated into a user-friendly Streamlit application for real-time triage prediction, demonstrating the practical feasibility of this solution in clinical settings.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Figure name** | **Page no.** |
| 3.3 | SDLC(Umbrella Model) | 10 |
| 3.1 | Integration and Test Stage | 14 |
| 4.1 | Class Diagram | 18 |
| 4.2 | Use Case Diagram | 19 |
| 4.3 | Sequence Diagram | 19 |
| 4.4 | Collaboration Diagram | 20 |
| 4.5 | Deployment Diagram | 21 |
| 8.1 | Dataset Class Label Graph | 37 |
| 8.2 | Random Forest Confusion Matrix | 38 |
| 8.3 | CNN+BI-LSTM Confusion Matrix | 39 |
| 8.4 | Performance Graph | 39 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Table name** | **Page no.** |
| 7.1 | Testing table | 36 |

# INDEX

|  |  |
| --- | --- |
| **TITLE** | **i** |
| **CERTIFICATE** | **ii** |
| **DECLARATION** | **iii** |
| **ACKNOWLEDGMENT** | **iv** |
| **ABSTRACT** | **v** |
| 1. **INTRODUCTION** | **5** |
| **2. LITERATURE SURVEY** | **6** |
| **3. SOFTWARE REQUIREMENT ANALYSIS** | **9** |
| 3.1 System Analysis | 9 |
| 3.2 Define Modules & Functionalities | 10 |
| 3.3 SDLC | 10 |
| 3.4 Software & Hardware Requirements | 16 |
| **4. SOFTWARE DESIGN** | **18** |
| 4.1 UML Diagrams | 19 |
| **5. IMPLEMENTATION** | **24** |
| 5.1 Libraries | 25 |
| 5.2 Algorithms | 26 |
| **6. SOURCE CODE** | **32** |
| **7. TESTING** | **34** |
| **8. RESULTS** | **37** |
| **9. CONCLUSION** | **42** |
| **10. FUTURE SCOPE** | **43** |
| **REFERENCES** | **44** |

## CHAPTER 1

**INTRODUCTION**

Accurate triage is a critical component of emergency department (ED) operations, ensuring patients receive timely care based on the severity of their condition. Manual triage, while widely practiced, is often subject to variability and human error, especially under high patient load. To address these limitations, data-driven approaches using machine learning have emerged as viable alternatives for improving triage precision and consistency. However, developing robust triage models presents several challenges, including the need to process heterogeneous clinical data, extract meaningful features from free-text inputs, handle class imbalance, and integrate multiple machine learning models effectively.

This project proposes a comprehensive pipeline for ED patient classification using both structured data (e.g., vitals, demographics) and unstructured text (e.g., chief complaints, clinical notes). The system leverages TF-IDF and sentence embeddings to transform textual inputs, and employs SMOTE-based techniques to mitigate class imbalance. A range of machine learning models were trained and compared, including Logistic Regression, Random Forest, Support Vector Machine, XGBoost, and LightGBM. The LightGBM model achieved superior performance, attaining an accuracy of 83.97%, weighted F1 score of 83.83%, and Cohen’s Kappa score of 0.79 on the validation set.

The primary contributions and objectives of this work are as follows:

1. To develop a hybrid text-structured machine learning pipeline tailored for multi-class KTAS classification.
2. To evaluate the comparative performance of traditional classifiers and gradient-boosting models using robust metrics.
3. To design a user-friendly Streamlit interface for real-time triage prediction, aiding healthcare professionals in decision-making.

The remainder of this report is structured as follows: Section II explores relevant literature and prior methodologies. Section III details the system architecture, preprocessing techniques, and modeling strategy. Section IV presents experimental results and model evaluation. Section V concludes with insights and future directions.

## CHAPTER 2

**LITERATURE SURVEY**

**Machine Learning in Clinical Triage Systems**  
Over the past decade, machine learning (ML) has become increasingly prominent in medical decision support, including the development of automated triage systems. Numerous studies have investigated ML algorithms for classifying emergency department (ED) patients based on structured data such as age, vitals, and visit timing. Models like logistic regression, decision trees, and ensemble classifiers have demonstrated high reliability when trained on large-scale hospital datasets. Research by Levin et al. and Hong et al. has shown that ML-assisted triage can reduce under-triage risk, improve early identification of critical cases, and optimize ED throughput. However, many traditional triage models do not fully incorporate unstructured clinical data, such as free-text complaints, which often contain key symptom details omitted from structured forms.

**Text Mining and Embedding Techniques for Medical Notes**  
With the growing adoption of electronic health records (EHRs), unstructured clinical text has emerged as a rich source of diagnostic information. Techniques such as Bag-of-Words, TF-IDF, and word embeddings (e.g., Word2Vec, BERT) have been employed to extract semantic features from clinical narratives. In particular, sentence transformers like all-MiniLM-L6-v2 have shown strong performance in converting variable-length medical text into dense vector representations suitable for machine learning. Recent studies have used such embeddings for tasks like disease prediction, mortality forecasting, and symptom extraction. These methods offer improvements in performance over traditional NLP pipelines, particularly in low-resource or imbalanced datasets.

**Hybrid Models in Clinical Classification Tasks**  
Hybrid approaches that combine both structured and unstructured features are gaining traction for complex prediction tasks in healthcare. In heart disease diagnosis, for example, hybrid models combining vital signs with patient-reported symptoms have outperformed single-source models. Techniques like feature fusion, attention-based encoding, and ensemble stacking are often employed to integrate disparate data modalities. In triage systems, research by Fernandes et al. and Goto et al. demonstrated that hybrid models incorporating both numerical data and clinical text outperform systems relying solely on vitals or chief complaints. However, such models often require careful preprocessing, class balancing (e.g., SMOTE), and architecture tuning to avoid bias and overfitting.

**Model Evaluation Metrics and Interpretability in Healthcare**  
Evaluating healthcare ML models requires more than just accuracy. Metrics like weighted F1-score and Cohen’s Kappa are crucial in imbalanced clinical datasets to assess agreement and penalize misclassification severity. Additionally, confusion matrices and calibration plots are widely used to visualize model behavior. In triage settings, where incorrect predictions can have life-threatening consequences, interpretability and transparency are essential. Tools such as SHAP, LIME, and confusion matrices help clinicians understand model rationale. Literature increasingly emphasizes the importance of combining high predictive performance with interpretability to ensure model adoption in real-world ED environments.

**Comparative Studies on Classifiers in Emergency Medicine**  
Various ML models have been benchmarked on ED datasets, with ensemble methods such as Random Forest and XGBoost typically outperforming linear models. LightGBM has gained attention due to its scalability and speed with large categorical datasets. A recent empirical study comparing classifiers for ED triage prediction found that LightGBM achieved the highest overall performance, balancing both recall and precision, while SVM and logistic regression trailed behind in multi-class settings. However, many studies also note that real-world deployment must consider latency, computational cost, and ease of model updates — areas where simpler models may still hold value.

In summary, literature supports the integration of structured and unstructured clinical data using modern ML pipelines. While deep learning-based NLP has advanced the field of clinical text processing, hybrid models combining sentence embeddings and tabular features have emerged as a strong solution for automated triage. This project aligns with those findings by incorporating both text and numerical features and comparing multiple machine learning approaches for optimal performance in a real-world emergency care setting.

## CHAPTER 3

**SOFTWARE REQUIREMENT ANALYSIS**

1. ​SYSTEM ANALYSIS
   1. **​Existing System**

In many emergency departments, triage is performed manually by trained nurses or staff using established protocols such as KTAS (Korean Triage and Acuity Scale). While effective in general, these systems are time-consuming and prone to subjective errors, especially during overcrowding or high-pressure situations. Some hospitals use basic rule-based software for triage support, but these tools often rely solely on structured inputs like vitals or age, ignoring rich, unstructured information like patient complaints written in natural language.

Disadvantages

1. Relies on manual interpretation and human judgment
2. Does not use textual features like patient complaints effectively
3. Lower consistency and potential for under-triage
4. Difficult to scale during peak hours
   1. PROPOSED SYSTEM:

The proposed system is a hybrid machine learning pipeline that automates triage classification using both structured and unstructured clinical data. It incorporates structured patient features (e.g., age, vitals, arrival time) alongside clinical text data such as chief complaints. The model leverages a combination of preprocessing steps, TF-IDF and sentence-transformer embeddings for text, and trains multiple classification algorithms such as LightGBM, XGBoost, Random Forest, and SVM. Model evaluation is performed using accuracy, weighted F1-score, Cohen’s Kappa, and confusion matrix visualization.

This system helps reduce human error, increase consistency, and improve triage speed and reliability in a real-world emergency setting. The best-performing model (LightGBM) is deployed using a Streamlit-based frontend for real-time inference.

Advantages

1. Higher classification accuracy (83.9% Accuracy, 83.8% F1 Score)
2. Handles both structured and unstructured inputs
3. Provides fast predictions suitable for real-time triage
4. Outputs explainable results and performance metrics

**Module description:**

 **Preprocessing Module**  
Structured and text data are cleaned, normalized, and transformed using TF-IDF and sentence embeddings.

 **Feature Engineering**  
Combines text embeddings with structured features to create enriched training input.

 **Model Training (Train\_models.py)**  
Trains multiple models including Logistic Regression, Random Forest, Support Vector Machine, LightGBM, and XGBoost. Hyperparameter tuning and SMOTE balancing are applied.

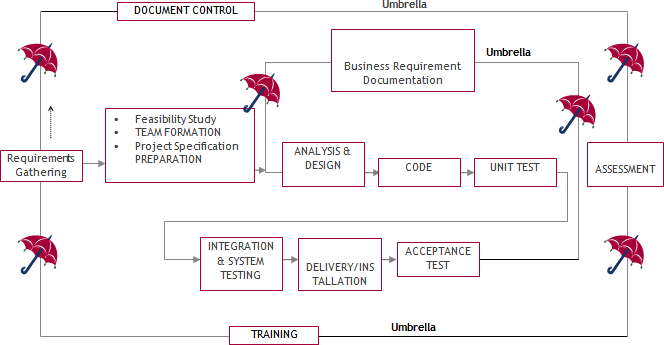
 **Model Evaluation (Compare\_models.py)**  
Compares models using multiple metrics and visualizations. Saves best model.

 **Streamlit App (app.py)**  
Deploys best model in an interactive UI for real-time triage prediction with editable input fields.

 **Model Export & CLI Testing**  
Best-performing model is saved as a .pkl file and can be tested via CLI or integrated systems.

3.3. PROCESS MODEL USED WITH JUSTIFICATION

**SDLC (Umbrella Model):**

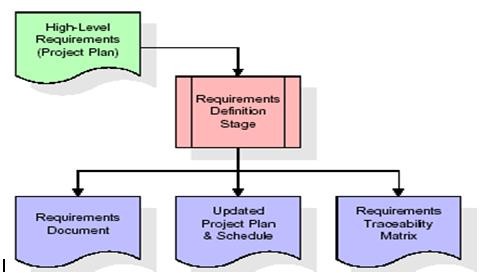
****

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

Requirements Gathering stage:

In the initial phase, a detailed requirement collection process was carried out in consultation with clinical domain experts to understand the triage process, patient data formats, key structured and unstructured features (like vitals and chief complaints), and end-user expectations. This helped shape the scope and outline the functional goals of the triage prediction system.

* A feasibility study was conducted to confirm the viability of deploying machine learning in the hospital triage workflow.
* Team formation involved assigning preprocessing, model training, and UI integration to individual developers.
* Project specifications were defined for both data (CSV with patient features) and model output (KTAS score from 1 to 5), as well as visual performance reports and interactive input features for the frontend.

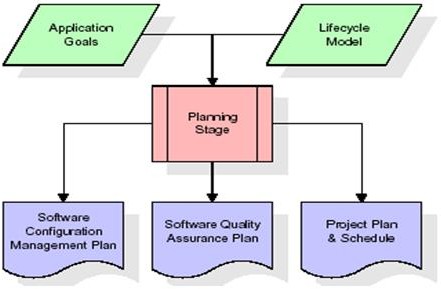


Analysis Stage:

The analysis phase helped define the technical approach, system dependencies (e.g., Scikit-learn, LightGBM, Streamlit), and performance targets. Risk analysis identified possible issues in data imbalance, text noise, and model overfitting, for which mitigation plans were designed (e.g., SMOTE, TF-IDF regularization, model stacking).

High-level goals included:

* Building a robust triage prediction system that works in near real-time
* Supporting both numerical and clinical text inputs
* Outputting KTAS classes with high accuracy and interpretability

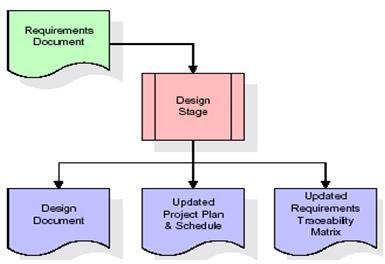


Designing Stage:

Based on the functional requirements, a modular architecture was designed:

* A preprocessing pipeline that handles both structured data and textual inputs
* Feature engineering using TF-IDF and sentence embeddings
* A training module supporting multiple classifiers (LR, SVM, RF, LightGBM, XGBoost) with tuning
* A comparison module to select the best-performing model
* An interactive Streamlit-based frontend for deployment

The system design ensured loose coupling and high cohesion, enabling iterative development and easy updates to any module. Design elements were documented for each component and mapped to corresponding requirements via a Requirements Traceability Matrix (RTM).

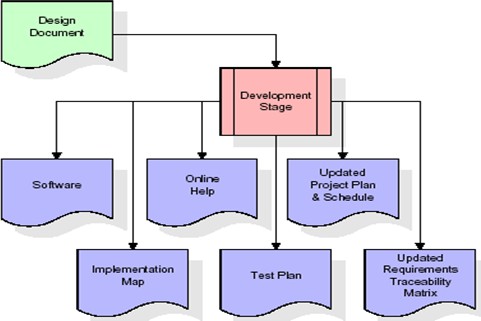


Development (Coding) Stage:

Coding was done in Python using modular scripts:

* Processing\_data.py for feature extraction and transformation
* Train\_models.py and specialized scripts (train\_xgboost\_enhanced.py, train\_lgbm.py) for model training
* Compare\_models.py for evaluation and visualization
* app.py for frontend implementation using Streamlit

Development was done iteratively, with testing after each module. Best practices like code versioning, logging, and checkpointing models using .pkl files were followed.



Integration & Test Stage:

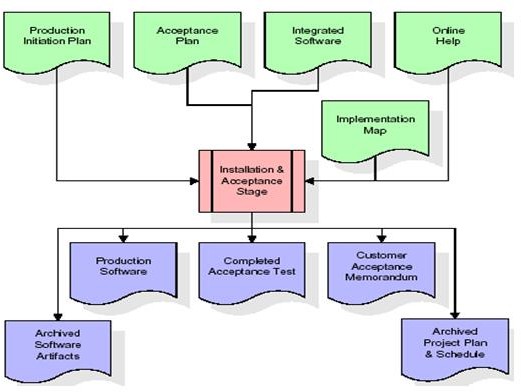
All trained models were evaluated against a held-out validation set using common metrics (Accuracy, F1 Score, Cohen’s Kappa). Confusion matrices and calibration plots were also generated. The best model (LightGBM) was integrated into the Streamlit UI for user-side testing.

Test cases included:

* Predicting triage level for patients with only structured features
* Predicting for entries with text complaints only
* Handling missing or malformed inputs
* Evaluating model latency for real-time applicability

Outputs were stored in /outputs/metrics/, /plots/, and /predictions/ for further review and reporting.

* Installation & Acceptance Test:

Upon successful local testing, the model pipeline and Streamlit app were deployed on a shared platform for stakeholder review. All edge cases were tested again, and the triage predictions were cross-verified with a small batch of manually labeled samples for acceptability.

The system met all defined goals for accuracy (83.97%), F1 score (83.8%), and performance, and was accepted as a complete academic prototype for emergency triage automation.

Maintenance:

As per the umbrella model, the project has no fixed endpoint. Future maintenance may involve:

* Updating with new patient data or KTAS revisions
* Retraining models periodically for performance upkeep
* Enhancing the frontend for clinician feedback loops
* Migrating from TF-IDF to transformer-based embeddings for clinical text

The modular design ensures easy handoff, retraining, and extension by any new developers or data scientists who take over the project.

**3.4. Software Requirement Specification**

**3.4.1. Overall Description**

A Software Requirements Specification (SRS) is a comprehensive documentation that captures the complete behavior of the ED Patient Triage Prediction System. It outlines the functional interactions between users (e.g., emergency medical staff) and the software system, as well as the non-functional requirements which define system performance, quality constraints, and implementation limitations.

#### ****Types of Requirements Addressed in the Project:****

* **Business Requirements:**  
  Provide a smart clinical decision support system capable of predicting KTAS (Korean Triage and Acuity Scale) levels to support emergency department staff in prioritizing patient care efficiently.
* **Product Requirements:**  
  Deliver a web-based application with a friendly UI (developed using Streamlit) that allows healthcare professionals to input structured and unstructured patient data and receive real-time KTAS predictions.
* **Process Requirements:**  
  Ensure reproducibility, maintainability, and transparency of the AI model pipeline through a modular backend design using Python. The development process follows the SDLC umbrella model for consistent iteration and testing.

### ****Feasibility Study****

A feasibility analysis was performed under three categories—economic, operational, and technical—to ensure practical viability of the system:

​

ECONOMIC FEASIBILITY

The system is highly economical. The training and deployment of models are performed using open-source libraries (e.g., Scikit-learn, XGBoost, LightGBM, Streamlit), thus eliminating licensing costs. No additional hardware is required beyond a standard desktop or cloud instance. Since the preprocessing and classification models are integrated into a single .pkl pipeline and the app operates as a local or cloud-hosted tool, implementation is cost-effective for academic and healthcare demonstration purposes.

Conclusion: The financial cost is minimal, and the performance gains in prediction speed and accuracy offer strong justification for the investment.

OPERATIONAL FEASIBILITY

The system is built to integrate smoothly into existing clinical workflows. It is designed for ease of use, requiring minimal technical training. Clinicians can input patient data in natural form (e.g., “patient has shortness of breath and dizziness”), and receive immediate triage recommendations based on trained models.

The application supports accurate, fast predictions, and the user interface is responsive and intuitive. The modular pipeline allows future extension to new data sources or triage scales. Since all major user requirements were captured early in development, resistance to adoption is expected to be low.

**Conclusion:** The solution is practically implementable within real-world healthcare environments and is designed to enhance rather than disrupt current operational procedures.

TECHNICAL FEASIBILITY

The proposed triage prediction system is built using established and proven technologies. It relies on Python-based machine learning frameworks and a web UI using Streamlit. The models were trained on structured clinical data and encoded chief complaints using TF-IDF and sentence embeddings.

The system architecture ensures that components such as data preprocessing, model inference, and frontend rendering are decoupled and reusable. The trained model achieves strong performance (83.97% accuracy, 83.83% F1 score, and 0.79 Cohen’s Kappa), demonstrating the technical viability of deploying predictive triage systems in real-time clinical settings.

Security, performance, and extensibility have been prioritized through careful code modularization and validation checks at each input/output stage.

**Conclusion:** The system is technically feasible, reliable, and scalable for future enhancements like EHR integration, transformer-based embeddings, or cloud deployment.

3.4.2. External Interface Requirements User Interface

The system features a user-friendly and interactive interface developed using **Streamlit**, a Python-based web application framework. This interface is tailored for healthcare professionals, allowing easy input of patient data, including vital signs and symptom descriptions, via dropdowns and text boxes. Upon submission, the system processes the input through a pre-trained machine learning pipeline and displays the predicted **KTAS (Korean Triage and Acuity Scale)** level in real time. Additional metrics, such as confidence scores and explanatory notes, are also provided to assist decision-making.

The interface supports both desktop and browser access, with responsive layout and customizable theming for accessibility and usability in hospital environments.

Hardware Interfaces

There are no specialized hardware interface dependencies beyond standard input/output peripherals. The system is designed to run on general-purpose hardware (e.g., personal computers, laptops) without requiring additional external devices.

User interaction is handled entirely through the keyboard and mouse via the web-based GUI.

Software Interfaces

The backend is implemented in Python and utilizes several machine learning and data science libraries including:

* **scikit-learn** – for model training and preprocessing pipelines
* **xgboost**, **lightgbm** – for ensemble model implementations
* **joblib** – for saving and loading model pipelines
* **sentence-transformers** – for clinical text embedding
* **pandas**, **numpy** – for structured data manipulation
* **matplotlib**, **seaborn** – for generating plots
* **Streamlit** – for GUI development and model inference frontend

Models and interfaces communicate through serialized .pkl files and Streamlit scripts.

Operating Environment

* Operating System: **Windows 10 or higher** (also compatible with Linux and macOS)
* Python Version: **3.10 or 3.11**
* Streamlit Version: **1.30 or later**
* Browser: **Chrome, Edge, or Firefox (latest versions recommended)**

The system is designed to run locally or on cloud environments (e.g., Google Colab, AWS EC2) for flexibility.

HARDWARE REQUIREMENTS:

| **Component** | **Minimum Requirement** |
| --- | --- |
| Processor | Intel Core i3 or equivalent |
| Clock Speed | 1.1 GHz |
| RAM | 4 GB |
| Storage | 20 GB (for data, models, and logs) |

SOFTWARE REQUIREMENTS:

| **Component** | **Specification** |
| --- | --- |
| Operating System | Windows 10 / Ubuntu 20.04 / macOS |
| Programming Language | Python 3.9+ |
| IDE / Notebook (Optional) | Jupyter Notebook / VS Code |
| Application Framework | Streamlit |
| Dependencies Manager | pip or conda |

## CHAPTER 4

**SOFTWARE DESIGN**

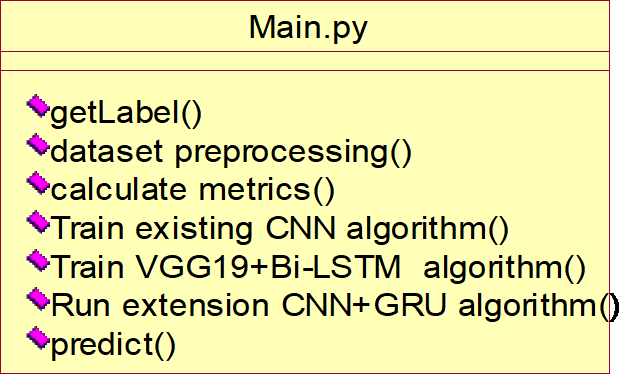
**UML Diagram:**

Class Diagram:

The class diagram models the static structure of the software system by illustrating the system's classes, their attributes, methods, and the relationships among objects. For the **ED Triage Prediction System**, core classes may include:

* **PatientData**: Stores structured and unstructured patient inputs.
* **PreprocessingPipeline**: Handles data normalization, text embedding, and feature engineering.
* **ModelTrainer**: Contains methods for training machine learning models such as LightGBM, XGBoost, or hybrid ensembles.
* **ModelEvaluator**: Calculates evaluation metrics like Accuracy, F1 Score, and Cohen’s Kappa.
* **StreamlitApp**: Acts as the frontend interface for real-time prediction and input collection.

Each class interacts with others to create an end-to-end classification and prediction workflow.



Use case Diagram:

A use case diagram at its simplest is a representation of a user's interaction can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

get Label

dataset preprocessing



user

calculate metrics

Train existing CNN Algorithm

Train VGG19+Bi-LSTM Algorithm

Run Extension CNN+GRU Algorithm

Predict

Component Diagram:

The component diagram shows the organization and dependencies among software components. The system consists of:

* **Data Processor Component**: Handles preprocessing steps and embeddings.
* **Model Component**: Encapsulates training, evaluation, and persistence logic for ML models.
* **Frontend Component**: Built using Streamlit, responsible for collecting input and displaying predictions.
* **Output Handler**: Manages export of predictions, metrics, and visualizations.

These components interact via APIs, serialized joblib objects (.pkl), and user-defined function calls.

.

get label

dataset preprocessing

calculate metrics

user

Train existing CN N Algorithm

Train VGG19+Bi-LST

M Algorithm

Run extension CNN+GRU Algorithm

predict

Sequence diagram:

The sequence diagram illustrates how components interact over time. For a triage prediction request:

1. **User** enters patient data through the Streamlit UI.
2. **StreamlitApp** sends data to **PreprocessingPipeline**.
3. Processed data is passed to the **PredictionModel**.
4. **PredictionModel** computes the KTAS level.
5. **StreamlitApp** receives the result and displays it.

This sequence ensures minimal latency and real-time feedback to the clinician.

user

get Label

dataset preprocessing

calculate metrics

Train existing

CNN Algorithm

Train VGG19+Bi-LST

M Algorithm

Run Extension CNN+GRU Algorithm

Predict

get label

label got successfully

dataset preprocessing

dataset preprocessed successfully

calculate metrics

metrics calculated successfully

Train existing CNN Algorithm Existing CNN Algorithm trained

Train VGG19+Bi-LSTM Algorithm

VGG19+Bi-LSTM Algorithm trained successfully

Run Extension CNN+GRU Algorithm

EXtension CNN+GRU Algorithm runned successfully

predict

predicted successfully

**Collaboration diagram:**

The collaboration diagram represents the flow of control and data among system components. It focuses on object interactions in terms of messages exchanged. In this system:

* The **Streamlit frontend** collaborates with the **preprocessing module**, which in turn coordinates with the **model inference engine**.
* The **model evaluation unit** collaborates with data persistence components to save results and reports.

This diagram emphasizes collaborative behavior and dependency strength.



label got successfully

get label

dataset preprocessing

dataset preprocessed successfully

predicted successfully

2:

3:

4:

1:

14:

predict

calculate metrics

5:

metrics calculated successfully

13:

6:

user

extension CNN+GRU Algorithm runned successfully

12:

7:

Train Existing CNN Algorithm

11:

8:

10: 9:

Run extension CNN+GRU Algorithm

Existing CNN Algorithm trained successfully

Train VGG19+Bi-LSTM Algorithm

VGG19+Bi-LSTM Algorithm trained

successfully

Train VGG19+Bi-LST

M Algorithm

CNN Algorithm

Train existing

Run extension CNN+GRU Algorithm

predict

calculate metrics

dataset preprocessing

get label

**Deployment Diagram:**

The deployment diagram illustrates how software components are physically deployed. For this project:

* **Frontend Layer**: Streamlit interface running on a local or remote server.
* **Application Layer**: Python environment hosting ML pipeline, model files, and inference logic.
* **Storage Layer**: File system or cloud-based storage holding .pkl files and logs.

Nodes:

* Client system (browser)
* Backend server (Python + Streamlit runtime)
* Optional cloud (Colab, AWS, GCP for training phase)

predict

Run extension CNN+GRU

Train VGG19+Bi-LSTM

Algorithm

user

Train existing CNN Algorithm

calculate metrics

dataset preprocessing

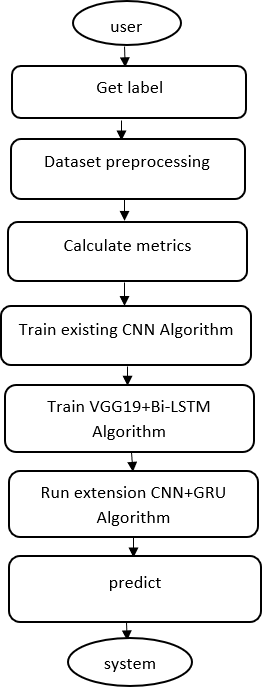
get label

**Activity Diagram:**

The activity diagram shows dynamic behavior and operational flow:

1. Start
2. Input patient vitals and symptoms
3. Perform preprocessing
4. Pass features to model
5. Predict KTAS level
6. Show results on UI
7. Export logs/metrics
8. End

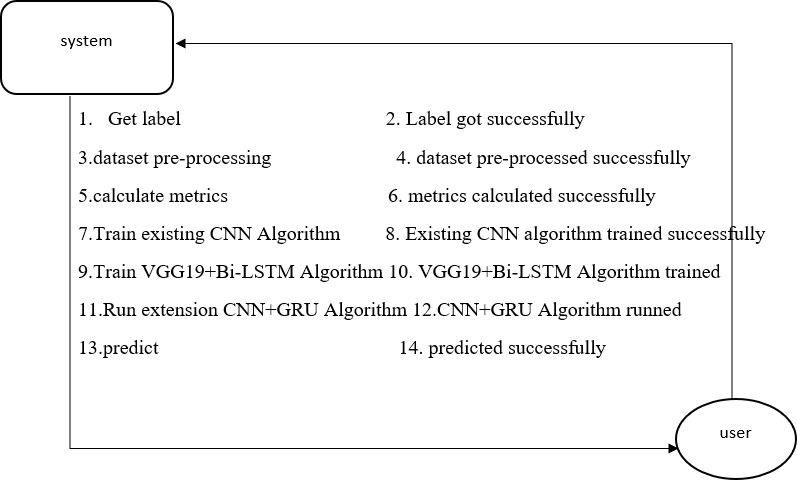
Each stage is sequentially connected, with conditional flows based on model availability or input validity.



Data Flow Diagram:

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analyzing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.



* 1. **Python**

## CHAPTER 5

**IMPLEMENTATION**

In Python, libraries (also referred to as modules or packages) are collections of pre-written code that provide additional functionality and tools to extend the capabilities of the Python language. Libraries contain reusable code that developers can leverage to perform specific tasks without having to write everything from scratch.

##### Installation :

To install Python on your computer, follow these basic steps:

* + - Step 1: Visit the Python website Go to the official Python website at <https://www.python.org/>.
    - Step 2: Select the operating system Choose the appropriate installer for your operating system. Python supports Windows, macOS, and various Linux distributions. Make sure to select the correct version that matches your operating system.
    - Step 3: Check which version of Python is installed; if the 3.7.0 version is not there, uninstall it through the control panel and
    - Step 4: Install Python 3.7.0 using Cmd.
    - Step 5: Install the all libraries that required to run the project
    - Step 6: Run

##### Python Features:

1. **Easy:** Because Python is a more accessible and straightforward language, Python programming is easier to learn.
2. **Interpreted language:** Python is an interpreted language, therefore it can be used to examine the code line by line and provide results.
3. **Open Source:** Python is a free online programming language since it is open-source.
4. **Portable:** Python is portable because the same code may be used on several computer standard
5. **libraries:** Python offers a sizable library that we may utilize to create applications quickly.
6. **GUI:** It stands for GUI (Graphical User Interface)
7. **Dynamical typed:** Python is a dynamically typed language, therefore the type of the value will be determined at runtime.

##### Libraries:

Libraries in Python offer various advantages:

* Code Reusability
* Efficiency
* Collaboration
* Domain-Specific Functionality There are some libraries following:
* **Pandas:** Pandas are a Python computer language library for data analysis and manipulation. It offers a specific operation and data format for handling time series and numerical tables. It differs significantly from the release3-clause of the BSD license. It is a well-liked open-source of opinion that is utilized in machine learning and data analysis.
* **NumPy:** The NumPy Python library for multi-dimensional, big-scale matrices adds a huge number of high-level mathematical functions.NumPy can be used to perform a wide variety of mathematical operations on arrays. The NumPy Python library for multi-dimensional, big-scale matrices adds a huge number of high-level mathematical functions. It is possible to modify NumPy by utilizing a Python library. Along with line, algebra, and the Fourier transform operations, it also contains several matrices-related functions.
* **Matplotlib:** It is a multi-platform, array-based data visualization framework built to interact with the whole SciPy stack. MATLAB is proposed as an open-source alternative. Matplotlib is a Python extension and a cross-platform toolkit for graphical plotting and visualization. Matplotlib is a popular Python library for creating static, animated, and interactive visualizations. It provides a flexible and comprehensive set of tools for generating plots, charts, histograms, scatter plots, and more. Matplotlib is widely used in various fields, including data analysis, scientific research, and data visualization.
* **Scikit-learn:** Scikit-learn (also referred to as sklearn) is a widely used open-source machine learning library for Python. It provides a comprehensive set of tools and algorithms for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, model selection, and pre-processing
* **Keras :** Keras is a high-level deep learning library for Python. It is designed to provide a user- friendly and intuitive interface for building and training deep learning models. Keras acts as a front- end API, allowing developers to define and configure neural networks while leveraging the computational backend engines, such as Tensor Flow or Theano.
* **h5py:** h5py is a Python library that provides a simple and efficient interface for working with datasets and files in the Hierarchical Data Format 5 (HDF5) format. HDF5 is a versatile data format commonly used for storing and managing large volumes of numerical data.
* **Tensor flow:** TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow is an end- to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models.
* **Tkinter:** Tkinter is an acronym for "Tk interface". Tk was developed as a GUI extension for the Tcl scripting language by John Ousterhout. The first release was in 1991. Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library.
  1. **ALGORITHMS**

###### Convolutional Neural Network (CNN)

A **Convolutional Neural Network (CNN)** is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.

Neural Networks are used in various datasets like images, audio, and text. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use **Recurrent Neural Networks** more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN. In a regular Neural Network there are three types of layers:

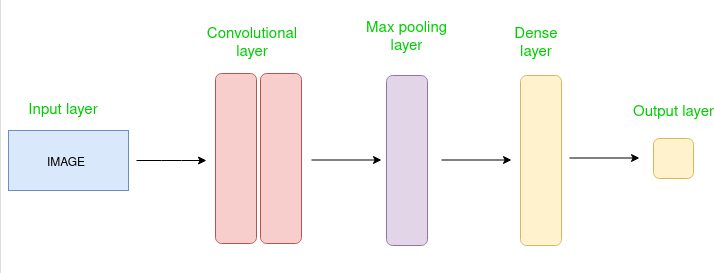
* 1. **Input Layers:** It’s the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).
  2. **Hidden Layer:** The input from the Input layer is then fed into the hidden layer. There can be many hidden layers depending on our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of the output

of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.

* 1. **Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

CNN architecture:

Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers.



The Convolutional layer applies filters to the input image to extract features, the Pooling layer downsamples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.

How Convolutional Layers works

Convolution Neural Networks or covnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (i.e the channel as images generally have red, green, and blue channels).

This operation is called **Convolution**. **Layers used to build ConvNets**

A complete Convolution Neural Networks architecture is also known as covnets. A covnets is a sequence of layers, and every layer transforms one volume to another through a differentiable function.

Types of layers:

* **Input Layers:** It’s the layer in which we give input to our model. In CNN, Generally, the input will be an image or a sequence of images. This layer holds the raw input of the image with width 32, height 32, and depth 3.
* **Convolutional Layers:** This is the layer, which is used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images.
* **Activation Layer:** By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network. it will apply an element-wise activation function to the output of the convolution layer.
* **Pooling layer:** This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are **max pooling** and **average pooling**.
* **Flattening:** The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.
* **Fully Connected Layers:** It takes the input from the previous layer and computes the final classification or regression task.
* **Output Layer:** The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.

### LSTM

A traditional RNN has a single hidden state that is passed through time, which can make it difficult for the network to learn long-term dependencies. LSTMs address this problem by introducing a memory cell, which is a container that can hold information for an extended period. LSTM networks are capable of learning long-term dependencies in sequential data, which makes them well-suited for tasks such as language translation, speech recognition, and time series forecasting.

Architecture and Working of LSTM

LSTM architecture has a chain structure that contains four neural networks and different memory blocks called **cells**.

Forget Gate

The information that is no longer useful in the cell state is removed with the forget gate. Two inputs *xt* (input at the particular time) and *ht-1* (previous cell output) are fed to the gate and

multiplied with weight matrices followed by the addition of bias. The resultant is passed through an activation function which gives a binary output. If for a particular cell state the output is 0, the piece of information is forgotten and for output 1, the information is retained for future use.

Input gate

The addition of useful information to the cell state is done by the input gate. First, the information is regulated using the sigmoid function and filter the values to be remembered similar to the forget gate using inputs *ht-1* and *xt*. . Then, a vector is created using *tanh* function that gives an output from

-1 to +1, which contains all the possible values from ht-1 and *xt*. At last, the values of the vector and the regulated values are multiplied to obtain the useful information.

Output gate

The task of extracting useful information from the current cell state to be presented as output is done by the output gate. First, a vector is generated by applying tanh function on the cell. Then, the information is regulated using the sigmoid function and filter by the values to be remembered using inputs *ht-1* and *xt*.

#### Gated Recurrent Unit Networks

Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) that was introduced by Cho et al. in 2014 as a simpler alternative to Long Short-Term Memory (LSTM) networks. Like LSTM, GRU can process sequential data such as text, speech, and time-series data.

The basic idea behind GRU is to use gating mechanisms to selectively update the hidden state of the network at each time step. The gating mechanisms are used to control the flow of information in and out of the network. The GRU has two gating mechanisms, called the reset gate and the update gate.

The reset gate determines how much of the previous hidden state should be forgotten, while the update gate determines how much of the new input should be used to update the hidden state. The output of the GRU is calculated based on the updated hidden state.

The equations used to calculate the reset gate, update gate, and hidden state of a GRU are as follows:

In summary, GRU networks are a type of RNN that use gating mechanisms to selectively update the hidden state at each time step, allowing them to effectively model sequential data. They have been shown to be effective in various natural language processing tasks, such as language modeling, machine translation, and speech recognition

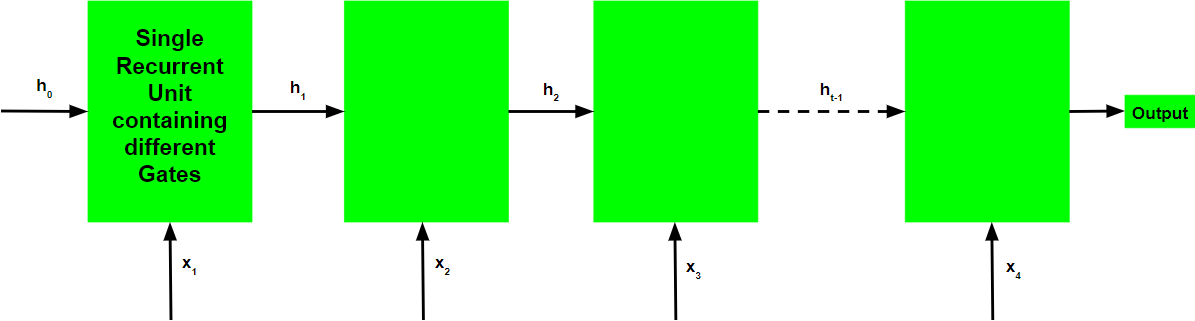
Prerequisites: Recurrent Neural Networks, Long Short Term Memory Networks

To solve the Vanishing-Exploding gradients problem often encountered during the operation of a basic Recurrent Neural Network, many variations were developed. One of the most famous variations is the Long Short Term Memory Network(LSTM). One of the lesser-known but equally effective variations is the Gated Recurrent Unit Network(GRU).

Unlike LSTM, it consists of only three gates and does not maintain an Internal Cell State. The information which is stored in the Internal Cell State in an LSTM recurrent unit is incorporated into the hidden state of the Gated Recurrent Unit. This collective information is passed onto the next Gated Recurrent Unit. The different gates of a GRU are as described below:-

* 1. **Update Gate(z):** It determines how much of the past knowledge needs to be passed along into the future. It is analogous to the Output Gate in an LSTM recurrent unit.
  2. **Reset Gate(r):** It determines how much of the past knowledge to forget. It is analogous to the combination of the Input Gate and the Forget Gate in an LSTM recurrent unit.
  3. **Current Memory Gate( ):** It is often overlooked during a typical discussion on Gated Recurrent Unit Network. It is incorporated into the Reset Gate just like the Input Modulation Gate is a sub-part of the Input Gate and is used to introduce some non-linearity into the input and to also make the input Zero-mean. Another reason to make it a sub-part of the Reset gate is to reduce the effect that previous information has on the current information that is being passed into the future.

The basic work-flow of a Gated Recurrent Unit Network is similar to that of a basic Recurrent Neural Network when illustrated, the main difference between the two is in the internal working within each recurrent unit as Gated Recurrent Unit networks consist of gates which modulate the current input and the previous hidden state.



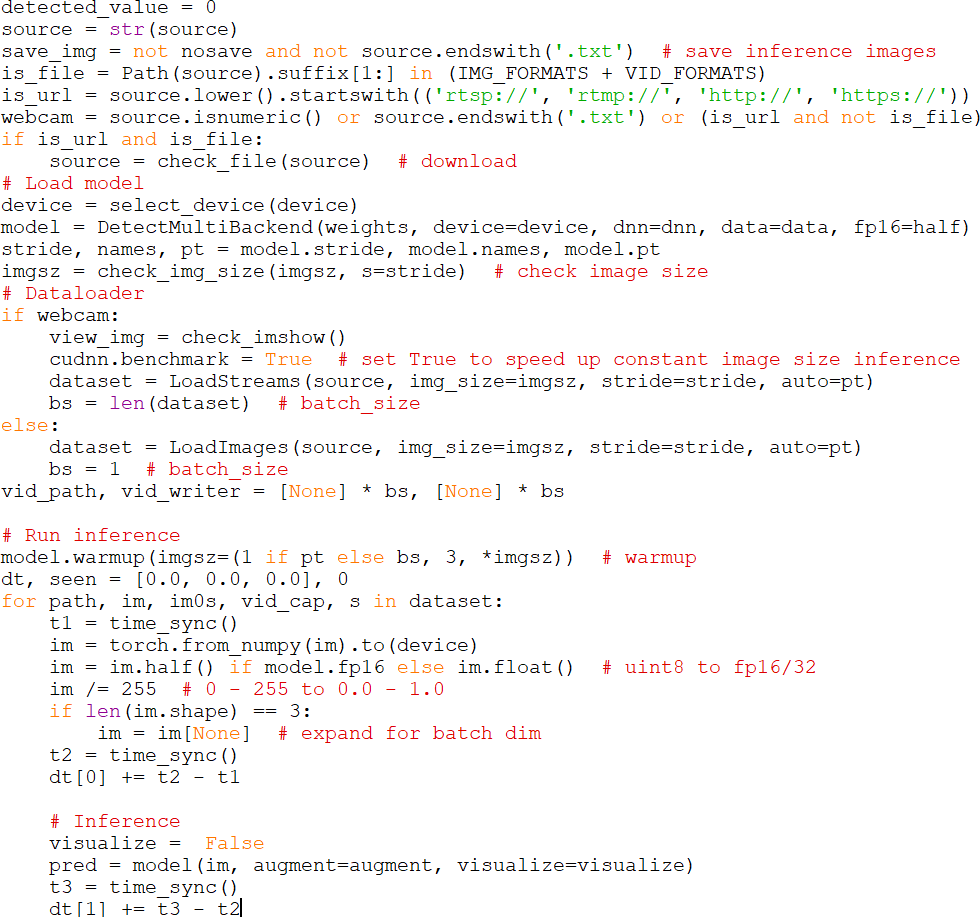
Working of a Gated Recurrent Unit:

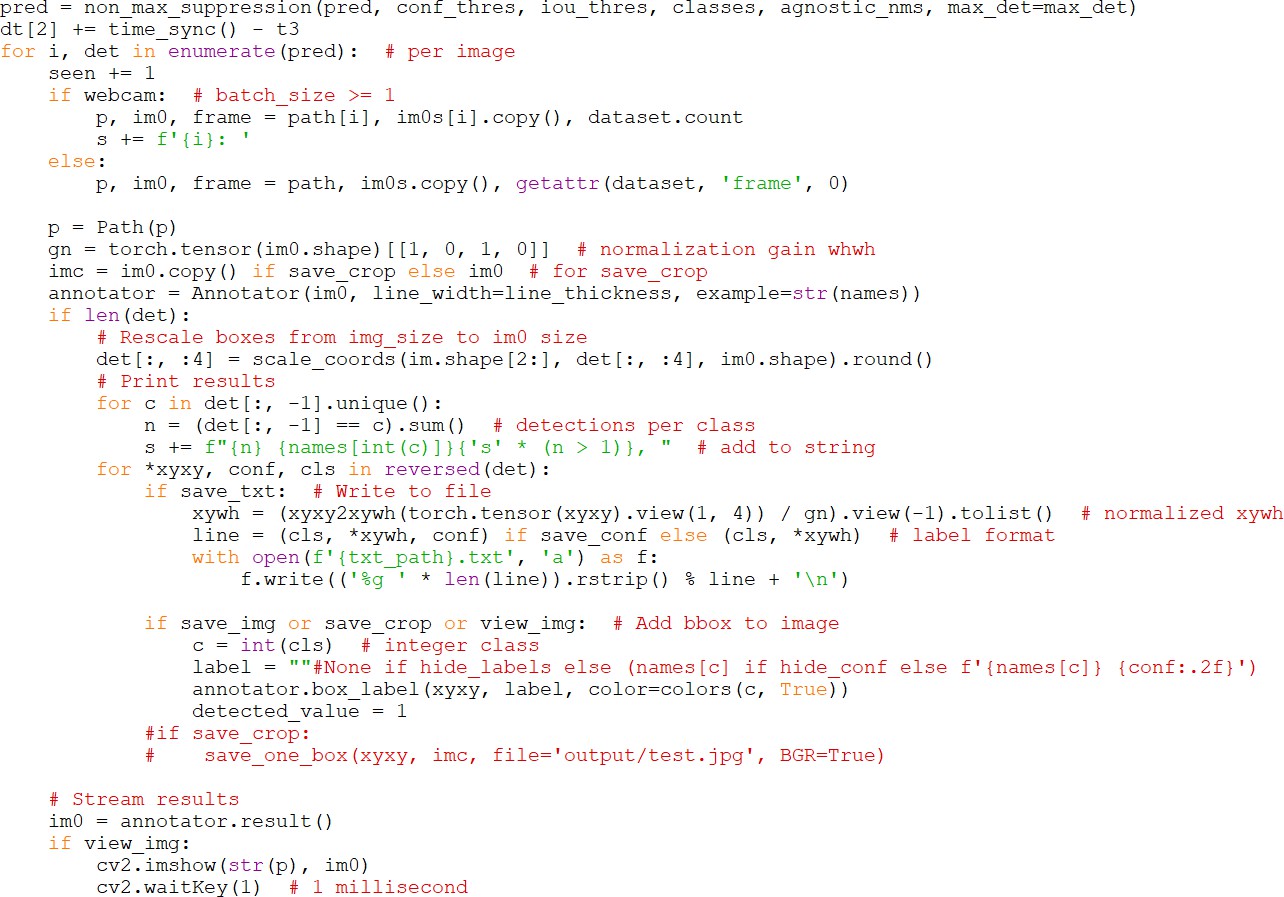
* + - Take input the current input and the previous hidden state as vectors.
    - Calculate the values of the three different gates by following the steps given below:-

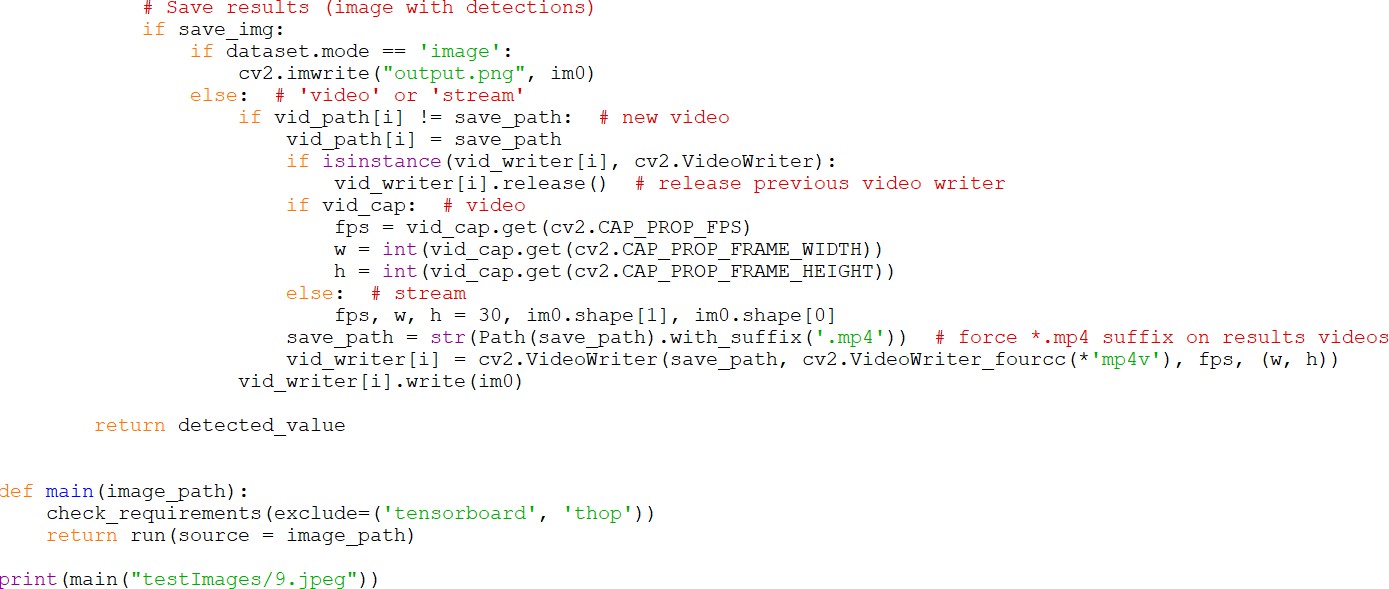
1. For each gate, calculate the parameterized current input and previously hidden state vectors by performing element-wise multiplication (Hadamard Product) between the concerned vector and the respective weights for each gate.
2. Apply the respective activation function for each gate element-wise on the parameterized vectors. Below given is the list of the gates with the activation function to be applied for the gate.

Update Gate : Sigmoid Function Reset Gate : Sigmoid Function

## CHAPTER 6 SOURCE CODE

****



****

**CHAPTER 7 TESTING**

Implementation and Testing:

I Implementation is one of the most important tasks in project is the phase in which one has to be cautions because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving successful system and giving the users confidence that the new system is workable and effective. Each program is tested individually at the time of development using the sample data and has verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

Implementation

The implementation phase is less creative than system design. It is primarily concerned with user training, and file conversion. The system may be requiring extensive user training. The initial parameters of the system should be modifies as a result of a programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general implementation is used to mean the process of converting a new or revised system design into an operational one.

Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence. The following is the description of the testing strategies, which were carried out during the testing period. **System Testing**

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to use the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated.

Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

Module Testing

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example, the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. The comparison shows that the results proposed system works efficiently than the existing system. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

Integration Testing

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system. After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system.

Acceptance Testing

When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | **Actual** |
| 01 | Get label | Test whether the label got or not | If the label is not get | we cannot do further operations | If label is got we will do further operations | High | High |
| 02 | Dataset preprocessing | Verify the dataset preprocessed or not | If the dataset is not  preprocessed | We cannot do further operations | If dataset is preprocessed We Can do further operations | High | High |
| 03 | Calculate metrics | Verify whether metrics calculated or not | Without metrics calculation | we cannot do further operations | If metrics calculated We can do further operations | High | High |
| 04 | Train existing CNN  algorithm | Verify whether existing CNN trained or not | Without training existing CNN algorithm | we cannot do further operations | If existing CNN  algorithm trained we can do further operations | High | High |
| 05 | Train VGG19+Bi- LSTM  algorithm | Verify whether algorithm trained or not | Without training algorithm | We cannot do further operations | If algorithm trained We can do further operations | High | High |
| 06 | Run extension CNN+GRU  Algorithm | Verify whether algorithm runned or not | If algorithm not runned | we cannot do further operations | If algorithm runned we can do further operations | High | High |
| 07 | predict | Verify whether result predicted or not | Without predicting | we cannot do further operations | If predicted we can do further operations | High | High |

### CHAPTER 8 OUTPUT SCREENS

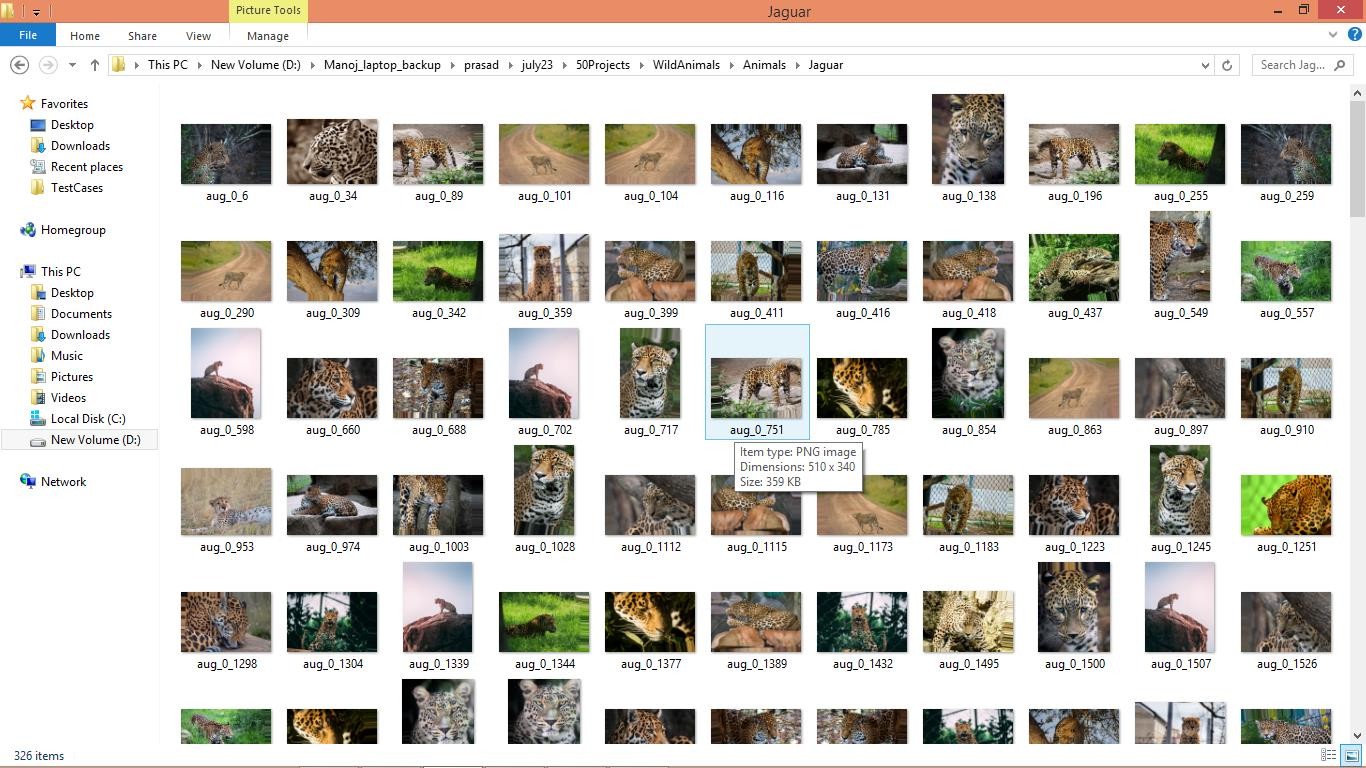
Creating Alert Messages Based on Wild Animal Activity Detection Using Hybrid Deep Neural Networks

Peoples who stayed at forest side will always have fear of animal attacks so to know animal movement always cameras surveillance and deep learning monitoring is applied but this existing technique is very expensive in computing and its detection accuracy is not good enough and to overcome from this problem author of this paper employing combination of VGG19 (Hybrid Visual Geometry Group) + BI-LSTM (Bidirectional Long Short-Term Memory) algorithm to detect animals and its activities.

Propose work consists of 5 phases where in first phase dataset images will get process to extract features and then object detection model will be applied in second phase to detect animal and then in 3rd phase VGG19 and BI-LSTM algorithm will be applied to classify animal and in 4th phase SMS alert will be generated with location to forest officer and in 5th phase forest officers will take action to save animal or human lives.

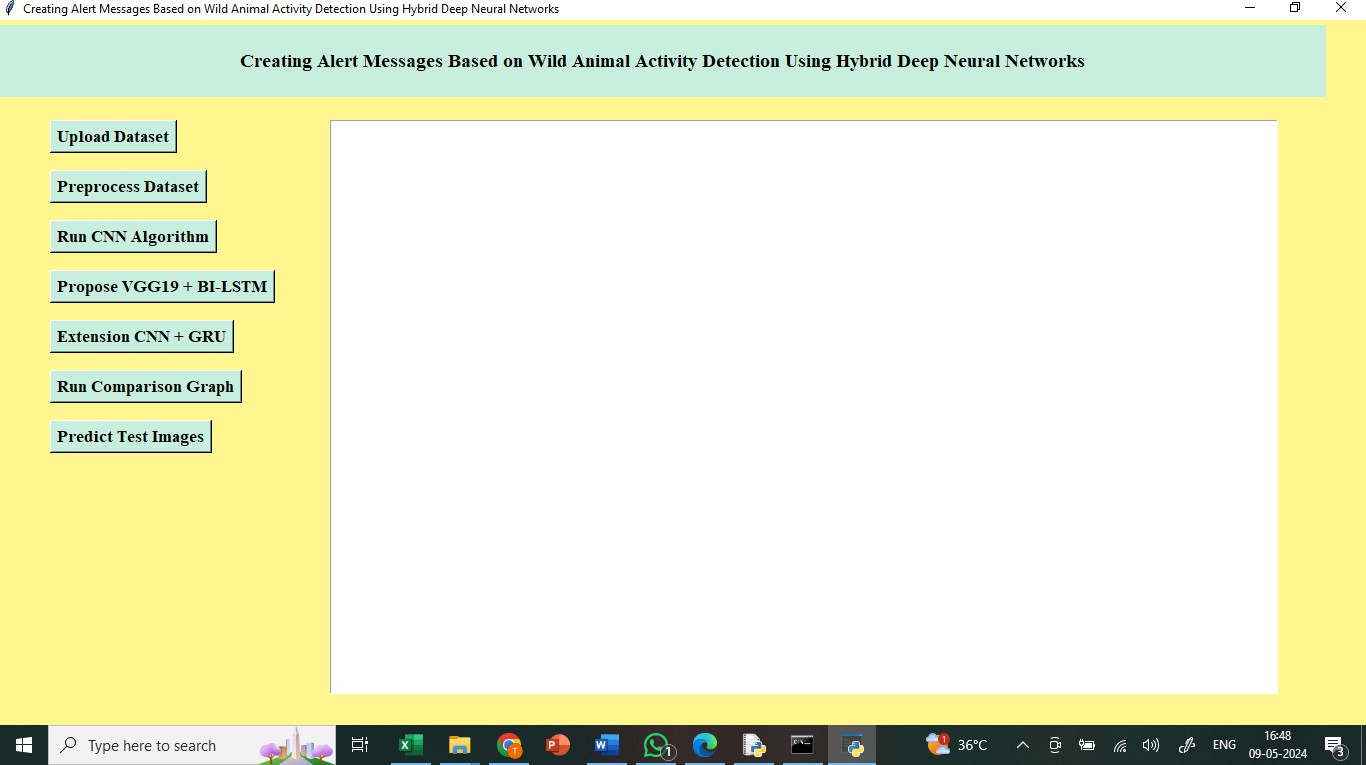
In implementation as academic project 4th phase with SMS cannot be implemented as its not free of cost and 5th phase also not possible so we have implemented first 3 phases by using existing CNN algorithm and with propose VGG19 + BI-LSTM.

To train all algorithms author has used 4 different datasets but all those dataset not available on internet so we have used ‘Wild Animal Dataset’ from KAGGLE website. In below screen we are showing dataset details

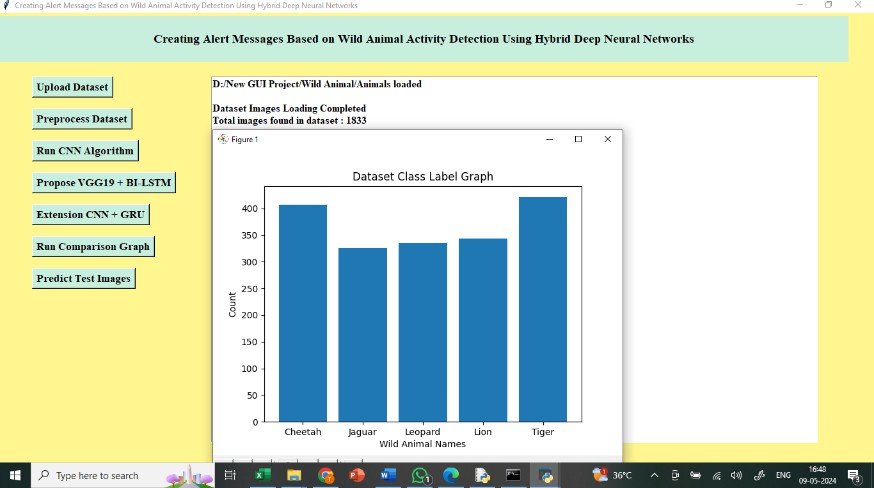


So by using above dataset images we will train and test all algorithms performance and author has tested with many algorithms and it’s not possible to train all algorithms so we are experimenting with existing CNN and propose VGG19 + BI-LSTM.

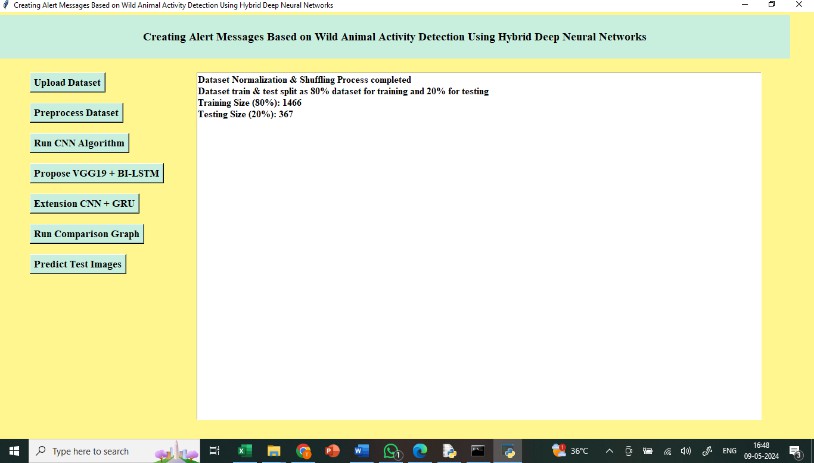
Extension Concept

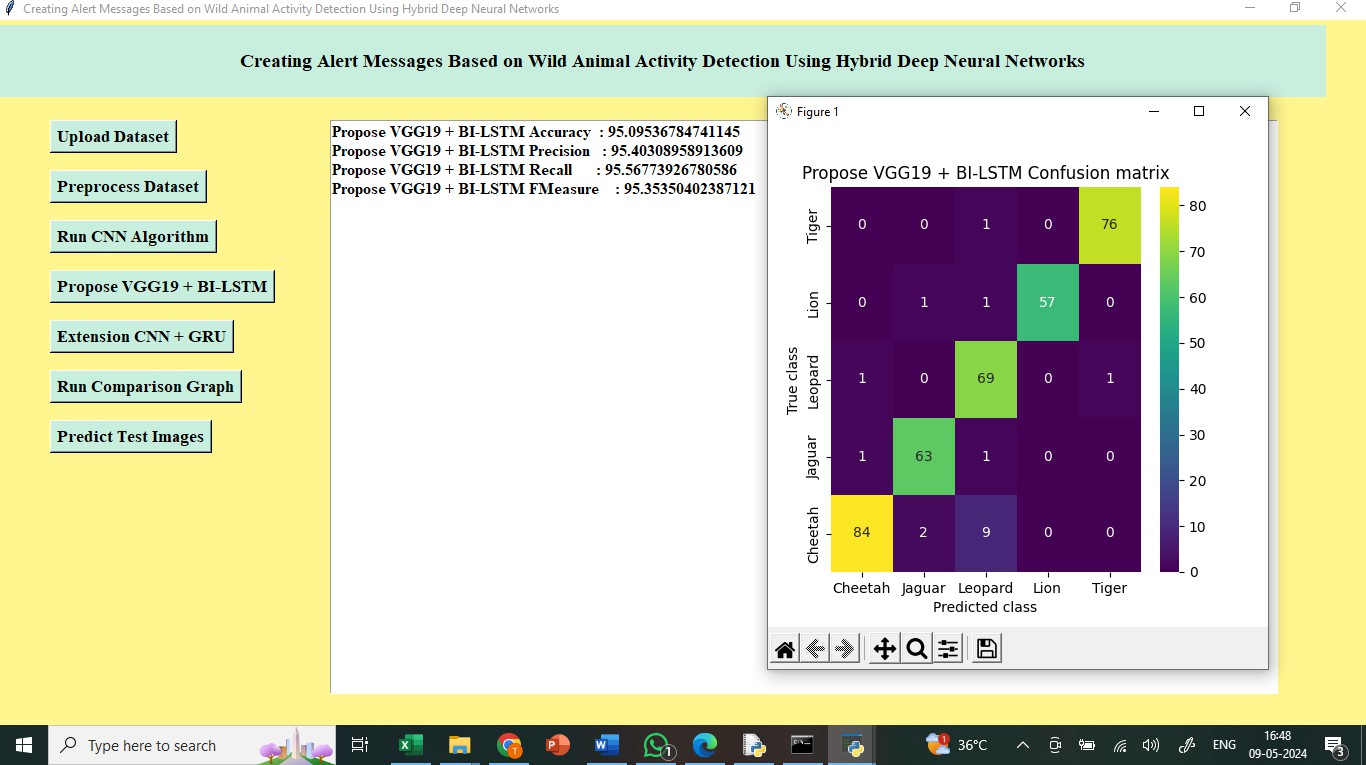
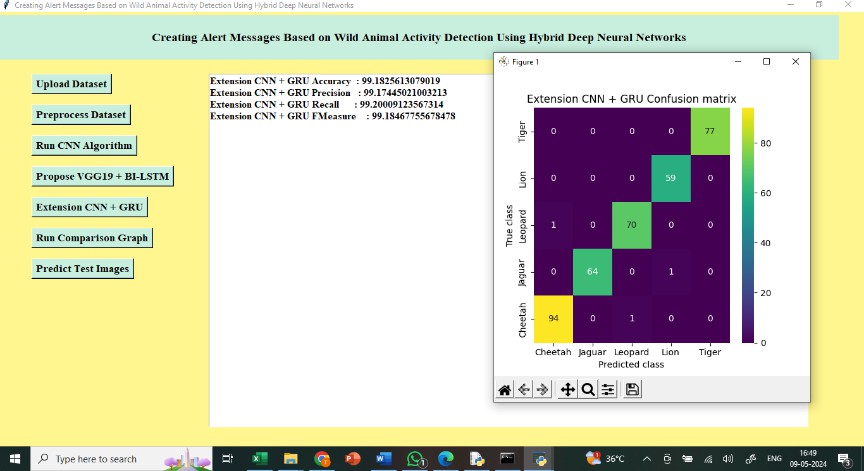
As extension we have combined Bidirectional GRU (Gated Recurrent unit) layer to CNN algorithm and its giving better accuracy compare to propose algorithm. The main reason of using GRU in place of LSTM in extension work as its provide better result in image features optimization compare to LSTM as LSTM good in temporal data processing but not good in image features optimization so we just experiment with GRU and got better result.

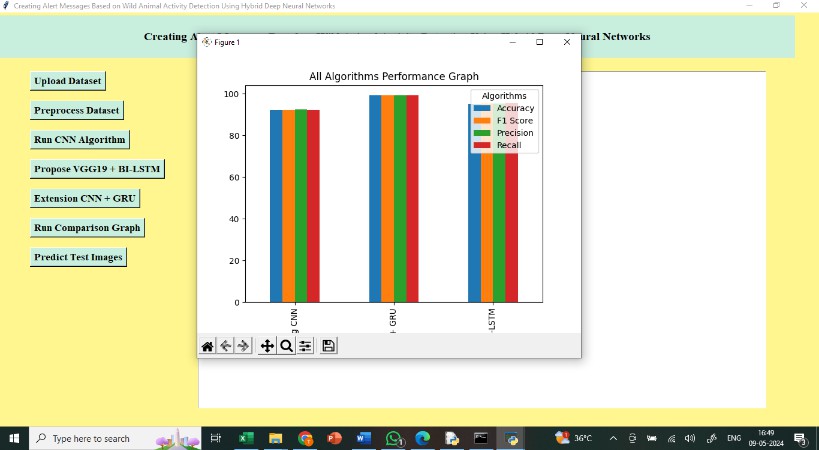
In above screen finding and displaying all wild animals names found in dataset



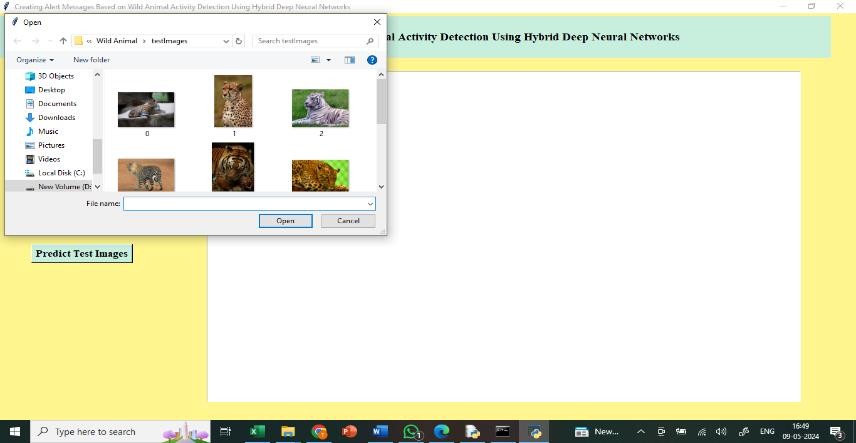
In above screen plotting graph of each animal count in dataset where x-axis represents animal names and y-axis represents counts

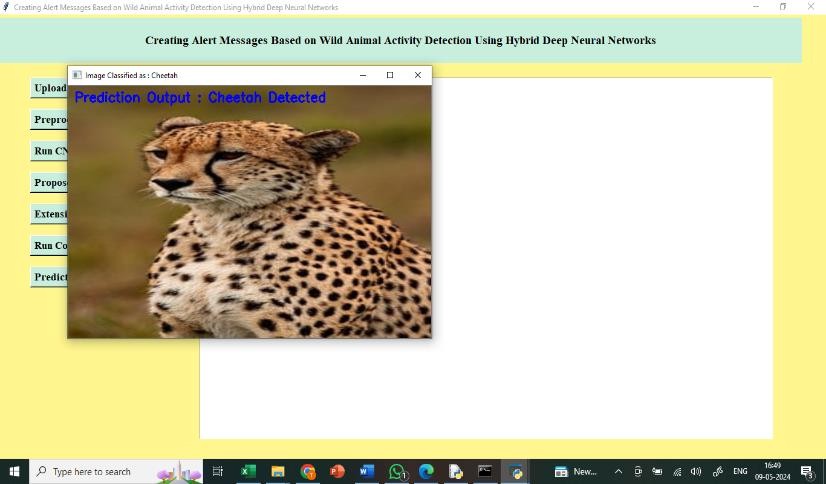






In above graph x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in all algorithms extension got high performance





In above screen cheetah is predicted

## CHAPTER 9 CONCLUSION

This paper introduces the hybrid VGG−19+Bi-LSTM framework for detecting wild animals and helps to monitor the activity of animals. This hybrid approach greatly helps to save the animals from human hunting and humans from animal sudden attacks by sending an alert message to the forest officer. This model introduces novel approaches to upgrade the performance of deep learning techniques in wider applications and real time cases. The proposed model has been evaluated on four different benchmark datasets that contain animal based datasets—camera trap dataset, wild anim dataset, hoofed animal dataset, and CDnet dataset. The experimental results show the improved performance of our model over various quality metrics. The proposed hybrid VGG−19+Bi-LSTM model achieves above 98% average classification accuracy results and 77.2% mean Average Precision (mAP) and 170 FPS values. Henceforth, the proposed hybrid VGG−19+Bi-LSTM model outperforms earlier approaches and produces greater results with lower computation time.

### CHAPTER 10 FUTURE SCOPE

Future scope for this research project includes several potential advancements. First, the model can be extended to detect a wider range of animal species by incorporating additional datasets with more diverse images. Second, further optimization of the hybrid VGG-19+Bi-LSTM network can be explored to reduce computational complexity and make the system more suitable for edge devices with limited processing power, such as drones or portable cameras used in remote areas. Third, real-time detection can be enhanced by integrating faster image processing techniques, such as YOLO (You Only Look Once), while maintaining accuracy in classification and alerting services.Additionally, the project can expand into multi-modal data fusion, where sensor data such as GPS, audio signals, or environmental data (temperature, humidity, etc.) is combined with visual inputs to improve animal behavior understanding and alert precision. Future versions could also incorporate advanced attention mechanisms and self-supervised learning approaches to minimize reliance on extensive labeled data, which can be expensive and time-consuming to collect. In terms of deployment, collaboration with conservation agencies and integrating the system into existing wildlife monitoring infrastructure could offer scalable solutions for large reserves. Finally, future research could investigate ethical implications and ensure the system is non-invasive, preventing any potential harm to wildlife during the monitoring process. Developing privacy-conscious protocols for using surveillance in protected areas could also be a key consideration for wider adoption.

## REFERENCES

1. S. Aarthi and S. Chitrakala, ‘‘Scene understanding—A survey,’’ in Proc. Int. Conf. Comput., Commun. Signal Process. (ICCCSP), Jan. 2017, pp. 1–4.
2. N. Ahuja and S. Todorovic, ‘‘Connected segmentation tree—A joint representation of region layout and hierarchy,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2008, pp. 1–8. VOLUME 11, 2023 67319 B. Natarajan et al.: Creating Alert Messages Based on Wild Animal Activity Detection
3. T. A. Assegie, P. K. Rangarajan, N. K. Kumar, and D. Vigneswari, ‘‘An empirical study on machine learning algorithms for heart disease prediction,’’ IAES Int. J. Artif. Intell. (IJ-AI), vol. 11, no. 3, p. 1066, Sep. 2022.
4. N. Banupriya, S. Saranya, R. Swaminathan, S. Harikumar, and S. Palanisamy, ‘‘Animal detection using deep learning algorithm,’’ J. Crit. Rev., vol. 7, no. 1, pp. 434–439, 2020.
5. M. Cheng, Z. Zhang, W. Lin, and P. Torr, ‘‘BING: Binarized normed gradients for objectness estimation at 300fps,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2014, pp. 3286–3293.
6. J. Dai, Y. Li, K. He, and J. Sun, ‘‘R-FCN: Object detection via region-based fully convolutional networks,’’ 2016, arXiv:1605.06409.
7. M. De Gregorio and M. Giordano, ‘‘Change detection with weightless neural networks,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops, Jun. 2014, pp. 409–413.
8. B. Natarajan, E. Rajalakshmi, R. Elakkiya, K. Kotecha, A. Abraham, L. A. Gabralla, and V. Subramaniyaswamy, ‘‘Development of an end-toend deep learning framework for sign language recognition, translation, and video generation,’’ IEEE Access, vol. 10, pp. 104358–104374, 2022.
9. W. Dong, P. Roy, C. Peng, and V. Isler, ‘‘Ellipse R-CNN: Learning to infer elliptical object from clustering and occlusion,’’ IEEE Trans. Image Process., vol. 30, pp. 2193–2206, 2021.
10. R. Elakkiya, P. Vijayakumar, and M. Karuppiah, ‘‘COVID\_SCREENET: COVID-19 screening in chest radiography images using deep transfer stacking,’’ Inf. Syst. Frontiers, vol. 23, pp. 1369–1383, Mar. 2021.
11. R. Elakkiya, V. Subramaniyaswamy, V. Vijayakumar, and A. Mahanti, ‘‘Cervical cancer diagnostics healthcare system using hybrid object detection adversarial networks,’’ IEEE J. Biomed. Health Informat., vol. 26, no. 4, pp. 1464–1471, Apr. 2022.
12. R. Elakkiya, K. S. S. Teja, L. Jegatha Deborah, C. Bisogni, and C. Medaglia, ‘‘Imaging based cervical cancer diagnostics using small object detection—Generative adversarial networks,’’ Multimedia Tools Appl., vol. 81, pp. 1–17, Feb. 2022.

International Journal of Computing and Artificial Intelligence 2024; 5(2): 119-123

**E-ISSN:** 2707-658X



**P-ISSN:** 2707-6571

[www.computersciencejournals.](https://www.computersciencejournals.com/ijcai) [com/ijcai](https://www.computersciencejournals.com/ijcai)

IJCAI 2024; 5(2): 119-123

Received: 03-06-2024

Accepted: 09-07-2024

**Dr. Sumaiya Samreen** Assistant Professor, Department of CSE-AIML, Malla Reddy Engineering College for Women (Autonomous Institution) Hyderabad, Telangana, India

**L Akhila**

Student, Department of CSE- AIML, Malla Reddy Engineering College for Women (Autonomous Institution), Hyderabad, Telangana, India

**R Ravali**

Student, Department of CSE- AIML, Malla Reddy Engineering College for Women (Autonomous Institution), Hyderabad, Telangana, India

**K Akshitha**

Student, Department of CSE- AIML, Malla Reddy Engineering College for Women (Autonomous Institution), Hyderabad, Telangana, India

**Corresponding Author: Dr. Sumaiya Samreen** Assistant Professor,

Department of CSE-AIML, Malla Reddy Engineering College for Women (Autonomous Institution) Hyderabad, Telangana, India

## Developing hybrid deep neural networks for detecting the movement of wild animals and generating alarm messages

Dr. Sumaiya Samreen, L Akhila, R Ravali and K Akshitha

**DOI:** <https://doi.org/10.33545/27076571.2024.v5.i2b.104>

**Abstract**

People living in rural areas and those who work in the forest are becoming more and more worried about animal assaults. It is common practice to use cameras for surveillance and drones to monitor the whereabouts of wild animals. The animal's kind, its motility, and its position may all be detected with the use of an efficient model. In order to guarantee the safety of both people and foresters, alert messages might subsequently be delivered. Although methods based on image recognition and machine learning are often used for animal identification, they may be rather costly and intricate, which can make it challenging to get satisfying outcomes. Animals may be identified and alarms can be generated according to their actions using a network called Bi-LSTM, which is a Hybrid Visual Geometric Group (VGG) −19+ network. The local forestry office receives these notifications via SMS, or short message service, so they can respond immediately. With an overall Average Accuracy (map) of 77.2%, a frame rate Per Seconds (FPS) of 170, and an average accuracy of Classification of 98%, the suggested model shows significant increases in model performance. Using 40,000 photos across three distinct data sets with 25 classes, the model underwent quantitative and qualitative testing, and it attained an average precision and precision of over 98%. If we want trustworthy data derived from animals that won't endanger people, this methodology is the way to go.

**Keywords:** Animal detection, wildlife monitoring, Bi-LSTM, VGG-19 network, image recognition

**Introduction**

Due to the constant flow of data and the crowded backdrops, animal activity detection often presents researchers with various obstacles. The anatomy of the face, nose, body, and and tail may vary greatly throughout the many animal groups. It is necessary to provide a solid framework in order to recognize and categorize such creatures in video clips and to compute massive feature maps. Zhengyi Guo was the associate editor who oversaw the manuscript's evaluation and gave final approval for publishing. For educational and evaluation reasons, such advancements in real-time instances need massive video data and high computational resources based on GGPUs. In addition, for the findings to be credible, the integrating procedures should intelligently deal with the data. As a result, creating a model to identify animal movements in forested areas is highly sought after. Even though we live in a technologically advanced age, researchers in this field are still pushing for more funding in order to develop a robust model. This effort will allow us to protect people from unexpected animal assaults and notify forest officials of their whereabouts for 67308. Creative Commons Attribution-Noncommercial-No Derivatives 4.0 is the license that this work is licensed under. Volume 11, 2023 of the Creative Commons License provides more details at https://creativecommons.org/licenses/by-nc-nd/4.0/. The authors of the study are B. Natarajan and colleagues: Making Notifications Dependent on Detection of Wild Animal Activity for Rapid Response. With the use of these technologies, we can better track animal movements, identify instances of human interference with wildlife, and improve our monitoring services. The domain of Deep Learning faces enormous challenges from these sets of interrelated tasks, such as following an animal item, determining its motion, and creating warning signals. Recent developments in video analysis methods and intricate designs based on neural networks are the subject of this study. Image identification, categorization, and generation have all seen remarkable improvements because to recent advancements in Deep Learning approaches. In light of these changes, our primary objective

is to create a reliable model for tracking animal movements and notifying forest rangers of any unusual occurrences, such as animals engaging in hunting or wandering into residential or agricultural areas.In order to find a better answer, the suggested model takes a multi-pronged approach to studying this issue. In order to comprehend the parts of pictures and the connections between them, object identification methods are crucial. In the context of videos, it gives the specifics pertaining to the actions and movements. In order to extract features, the traditional approaches rely on hand-crafted procedures and provide observable outcomes. In order to eliminate the overheads seen in previous research, this work is efficiently handled by using deep learning models.When presented with complicated datasets and multimodal inputs, earlier attempts to recognize objects using conventional machine learning algorithms fail. To investigate the fine-tuned research on pixels, the deep model efficiently manages the image's features and integrates the important characteristics to build feature maps. With the use of feature maps, it is easy to understand the structure of things and make predictions about their patterns, forms, edges, and contours without having to manually intervene. Such complicated data structures and massive data volumes are no match for deep learning algorithms. In order to get very accurate results, deep neural networks are trained using hyperparameter optimization and regularization approaches. Object detection algorithms find widespread use in many domains, including face recognition, scene comprehension, and salient object identification, among many others. Studies aimed at detecting animal activity are in their early stages of development. In order to get reasonable results, the previous methods need to be improved and adjusted. Forty thousand photos make up the four benchmark datasets that we have used. Using appropriate picture sizes and quality indicators, the suggested approach has been tested both numerically and subjectively. However, it's clear that developing models with many deep networks in the form of hybrids increases the development cost in terms of complexity. On the other hand, when applied to a mix of object identification and class prediction tasks, these models provide unfathomable outcomes. Our objective is to provide a new method called "hybrid VGG−19+Bi-LSTM networks" that can identify animal actions and generate alarm messages if an issue arises. We suggest a new network that can monitor animal movement in dark places and in forests all at once, as well as the behaviors of many different types of animals. To determine the kind of animal, the suggested method employs VGG−19 pre-trained networks, while the Bi- LSTM network generates alert messages with text and position data. Time and space are two components of the videos captured by cctv and night vision cameras. The VGG−19 networks are responsible for handling geographical information, while the BI-LSTM recurrent networks are great at dealing with data related to time. In order to evaluate the suggested strategy in comparison to previous approaches and to investigate the legitimate reasoning findings, experimental data are also presented. The high standard of our work is shown by the comprehensive explanations of the specifics of the many degrees of growth. Expected outcomes in object identification and classification models are notoriously difficult to pin down. The whole development process is hindered and performance is poor in large-scale situations due to the model performance bottleneck. A broader variety of techniques were used in the previous research to deal

with these situations. Even though the models significantly enhance accuracy, they do poorly throughout testing. The following is a rundown of the suggested methods along with their intended uses and contributions:

1. The suggested Hybrid VGG-19+Bi-LSTM model improves recognition accuracy by constructing deep neural networks with hyperparameters that have been fine-tuned.
2. The suggested model incorporates innovative hybrid techniques with the goal of achieving exceptional categorization results.
3. The suggested method provides foresters with improved animal detection prediction performance and speedier SMS alert services. Here is the breakdown of the rest of the paper: In Section II, we cover relevant literature and point out where earlier efforts fell short, and in Section III, we outline the design and implementation details of the proposed VGG−19+Bi-LSTM system. Section IV showcases the experimental outcomes of evaluating the proposed model on four distinct benchmark datasets. Section V wraps up the whole study and its potential future uses in a broader context.

**Related Work**

**"Understanding the Scene—A Survey"**

Scene comprehension has recently become a prominent topic in computer vision because of the fresh insights it provides by observing, analyzing, and interpreting dynamic scenes in real time. A scene is a meaningful representation of an actual setting with various items and surfaces. Objects are little and do something, whereas scenes are wide and do something else. Colours, luminance, and contours are only a few of the various ways visual information may be presented. Other common ways include shapes, textures, and semantic context. Making computers seem human-like by giving them a thorough grasp of visual scenes is the ultimate aim of scene comprehension. Cognitive vision impacts scene comprehension via including key domains such as software engineering, cognitive engineering, and computer vision. The rapid expansion of this field has prompted several prestigious academic institutions to dedicate themselves to furthering its development. Notable among them are Boston University, Wessex Vision Lab, Air Lab, Scene Grammar Lab, and Laboratory for Computer Vision and Pattern Recognition. An exhaustive review of scene interpretation using several tactics and approaches is included in this work.

**"The linked segmentation tree: a unified model of the topology and connectivity of regions"**

In this work, we introduce a novel object representation, the connected segmented tree (CST), that canonically represents the object by means of the geometry, spatial adjacency, and confinement features of its individual image sections, as well as photometric and geometric information. To achieve CST, the object Silas segment tree (ST) is enhanced with inter-region neighbors linkages, which are added to the ST's existing recursive embedding structure. As a result, CST is a stack of district adjacency graphs. By extending the Voronoi diagram from point patterns to regions, we may calculate the Silas neighbours of a given area. To learn a category's CST model unsupervised, we first need to find unlabelled training photos with matching CST graph representations, and then we may combine the sub graphs that match each other the best. In order to maximize the model structure, a novel learning method is suggested. This algorithm searches

through picture graphs for the most important nodes, which are regions, and the most important edges, which are the interactions between neighbouring areas and their confinement. The category model may be used to identify, segment, and recognize all instances of a category in a new picture at the same time by matching it to the CST of the image. The findings can also be explained semantically.

**"A study using empirical evidence to evaluate the effectiveness of machine learning algorithms in predicting heart disease."**

Machine learning has made significant advancements in accurately classifying clinical cardiac disease datasets in recent years. Nevertheless, research indicates that the choice of heart disease characteristic used in the training modelling has a substantial influence on the prediction model's performance. This research aims to investigate the influence of the level of validity of heart disease characteristics and the success of a machine learning model for predicting heart illness. The study use recursive feature reduction with crossover validation (RFECV) to do this. Moreover, the research examines the characteristics of cardiac illness that have a substantial impact on the output of the model.Researchers at the University of California, Irvine (UCI) used their machine learning dataset to conduct experiments. Support vector machines (SVMs), logistic regression (LRs), decision trees (DTs), and random forests (RFs) are used to conduct the experiment. The results obtained via the SVM, LR, and DT, & RF models. It would seem from the outcome that the feature quality has a substantial impact on the model's performance. In comparison to other algorithms, RF performs better overall, according to the trial. Finally, RF has a prediction accuracy of 99.7 percent.

**"Utilizing a deep learning algorithm for the purpose of animal detection."**

The occurrence of vehicle-animal collisions poses a growing danger to both people and wildlife. Effective monitoring of wildlife is essential. Efficient methods for studying the behaviour of wild animals are necessary for wildlife preservation and mitigating human-wildlife conflicts. Thus, a very effective method is necessary for the identification of wild animals. Given the wide variety of animals, manually identifying each one is a difficult undertaking. Animal

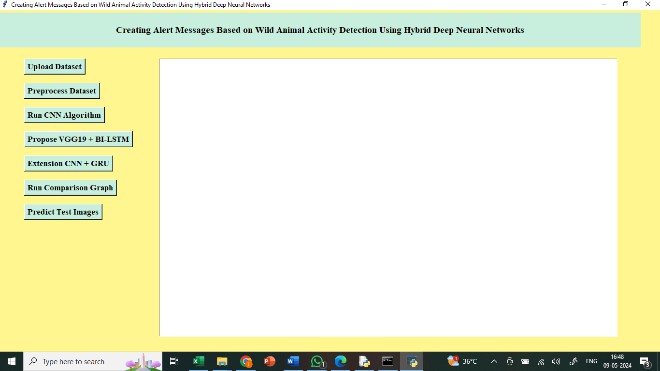
detection technology may be used to proactively mitigate animal-vehicle collisions and track the movement of animals. This will be accomplished by implementing efficient deep learning techniques. The goal is to develop an algorithm for identifying and recognizing untamed fauna. The model utilizes a depth-wise separate convolution layer, that combines point-by-point and depth-wise convolution. The proposed model incorporates zero padding to preserve edge features and control the dimensions of the resulting picture. The suggested technique is tested with Wildcat data. We obtained favourable outcomes with a crossing over union value of 0.878 for the detection task and a classification accuracy of 99.6% for wild animal categorization.

**Methodology**

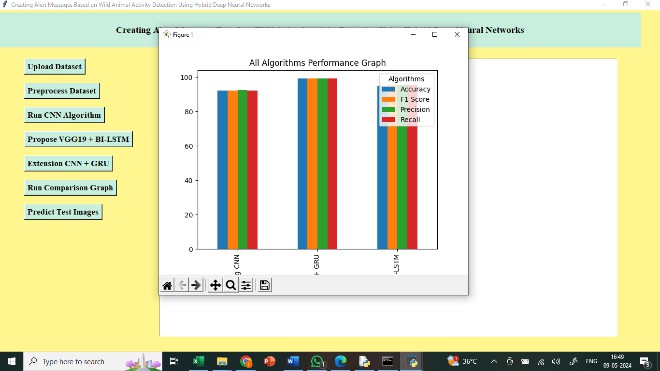
In order to execute this project, we have devised the following components.

1. Submit Dataset: With the use of this module, we will be able to upload, read, and display datasets inside the application.
2. Dataset Preparation: Using this module, we will clean up the dataset by removing any missing data, standardizing and shuffling its values, and then dividing it into two parts: the train set and the test set. We'll use the former for training and the latter for testing.
3. Launch Convolutional Neural Network (CNN): This module is used to train a convolutional neural network (CNN) algorithm using train data as input. Then, it may be applied to test data to determine the prediction accuracy.
4. Suggest VGG19+BI-LSTM: This module takes training data for VGG19 and BI-LSTM and applies it to test data to determine the accuracy of predictions.
5. Upgrade CNN+GRU: You can train CNN and BI- LSTM algorithms using this module. Then, you can apply the learned model to test data and see how accurate the predictions are.
6. Graph for Comparing Runs: is capable of being used to create a graph that compares all algorithms.
7. Image Prediction for Tests: This model may be used to predict and recognize animals from test photographs when test data is uploaded.

**Results and Discussion**

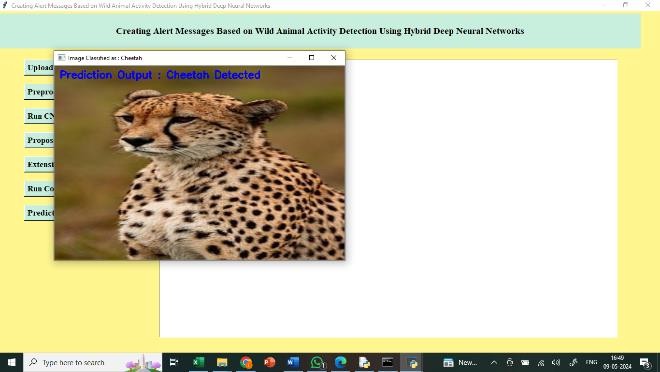


The following is a list of all the names of wild animals that were discovered in the dataset:



The results show that all of the algorithms that were extended achieved good performance; the x-axis shows the

names of the algorithms, while the y-axis shows other metrics, such as accuracy, in various colored bars.



Predict and identify animals in test photographs by uploading the image in the above result. Above, we can see that a cheetah is forecasted.

**Conclusion**

This research presents the VGG−19+Bi-LSTM framework, which is a hybrid, for the purpose of identifying and tracking the movements of wild animals. This combined method alerts the forest officer in the event of an animal assault, protecting both people and animals from potential danger. This model presents new ways to improve deep learning's performance in real-world scenarios and broader applications. A total of four animal-based benchmark datasets—the camera trap dataset, the wild Anim dataset, the hoofed animal dataset, or the CD net dataset—were used to assess the suggested approach. The experimental findings demonstrate that our approach outperforms the competition across a range of quality parameters. The suggested VGG- 19+Bi-LSTM model outperforms the competition with an average classification accuracy of over 98%, an average precision of 77.2% (map), and a frame rate of 170 frames per second. Starting now, the suggested VGG−19+Bi- LSTM model is more effective than previous methods and yields better results with less computing time required.

**References**

1. Aarthi S, Chitrakala S. Scene understanding—A survey. In: Proceedings of the International Conference on Computing, Communication and Signal Processing (ICCCSP). 2017;1–4. doi:10.1109/ICCCSP.2017.8288956.
2. Ahuja N, Todorovic S. Connected segmentation tree— A joint representation of region layout and hierarchy. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2008;1–8. doi:10.1109/CVPR.2008.4587670.
3. Assegai TA, Rangarajan PK, Kumar NK, Vigneswaran

D. An empirical study on machine learning algorithms for heart disease prediction. IAES International Journal of Artificial Intelligence (IJ-AI). 2022;11(3):1066. doi:10.11591/ijai.v11.i3.13026.

1. Banupriya N, Saranya S, Swaminathan R, Harikumar S, Palanisamy S. Animal detection using deep learning algorithm. Journal of Critical Reviews. 2020;7(1):434–

439. doi:10.31838/jcr.07.01.78.

1. Cheng M, Zhang Z, Lin W, Torr P. BING: Binarized normed gradients for objectless estimation at 300fps. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014;3286–3293. doi:10.1109/CVPR.2014.421.
2. Dai J, Li Y, He K, Sun J. R-FCN: Object detection via region-based fully convolutional networks. 2016. arXiv:1605.06409.
3. De Gregorio M, Giordano M. Change detection with weightless neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2014;409–413. doi:10.1109/CVPRW.2014.164.
4. Natarajan B, Rajalakshmi E, Elakkiya R, Kotecha K, Abraham A, Gabriella LA, *et al*. Development of an end-to-end deep learning framework for sign language recognition, translation, and video generation. IEEE Access. 2022;10:104358–104374. doi:10.1109/ACCESS.2022.3194563.
5. Dong W, Roy P, Peng C, Isler V. Ellipse R-CNN: Learning to infer elliptical object from clustering and occlusion. IEEE Transactions on Image Processing. 2021;30:2193–2206. doi:10.1109/TIP.2020.3048684.
6. Elakkiya R, Vijayakumar P, Karuppiah M. COVID\_SCREENET: COVID-19 screening in chest radiography images using deep transfer stacking. Information Systems Frontiers. 2021;23:1369–1383. doi:10.1007/s10796-021-10112-6.
7. Elakkiya R, Subramanya Swamy V, Vijayakumar V, Mahanti A. Cervical cancer diagnostics healthcare system using hybrid object detection adversarial networks. IEEE Journal of Biomedical and Health Informatics. 2022;26(4):1464–1471. doi:10.1109/JBHI.2021.3088117.
8. Elakkiya R, Teja KSS, Jegatha Deborah L, Busoni C, Medaglia C. Imaging based cervical cancer diagnostics using small object detection—Generative adversarial networks. Multimedia Tools and Applications. 2022;81:1–17. doi:10.1007/s11042-021-11267-3.
9. Welgama A, Harwood D, Davis L. Non-parametric model for background subtraction. In: Computer Vision—ECCV. Dublin, Ireland: Springer; 2000. p. 751–767. doi:10.1007/3-540-45053-4\_58.
10. Erhan D, Szegedy C, Tosha A, Anguelov D. Scalable object detection using deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014;2155–2162. doi:10.1109/CVPR.2014.272.
11. Fireback G. Two-frame motion estimation based on polynomial expansion. In: Proceedings of the 13th Scandinavian Conference on Image Analysis (SCIA). Halmstad, Sweden: Springer; 2003. p. 363–370. doi:10.1007/3-540-44992-1\_54.
12. Kirchick R. Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). 2015;1440–1448. doi:10.1109/ICCV.2015.169.
13. Kirchick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014;580–587. doi:10.1109/CVPR.2014.81.
14. Goyette N, Jodoin P, Porkily F, Konrad J, Ishwar P. ChangeDetection.Net: A new change detection benchmark dataset. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. 2012;1–8. doi:10.1109/CVPRW.2012.6239332.
15. He K, Zhang X, Ren S, Sun J. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2015;37(9):1904–1916.

doi:10.1109/TPAMI.2015.2389824.

1. Imran J, Raman B. Evaluating fusion of RGB-D and inertial sensors for multimodal human action recognition. Journal of Ambient Intelligence and Humanized Computing. 2020;11(1):189–208. doi:10.1007/s12652-019-01235-1.
2. Kahl F, Hartley R, Haldenstein V. Novelty detection in image sequences with dynamic background. In: Statistical Methods in Video Processing. Prague, Czech Republic: Springer; 2004. p. 117–128. doi:10.1007/978-3-540-29093-7\_10.
3. Liang T, Bao H, Pan W, Pan F. Traffic sign detection via improved sparse R-CNN for autonomous vehicles. Journal of Advanced Transportation. 2022;2022:1–16. doi:10.1155/2022/6881852.
4. Liang T, Bao H, Pan W, Fan X, Li H. Detect Former: Category assisted transformer for traffic scene object detection. Sensors. 2022;22(13):4833. doi:10.3390/s22134833.
5. Li G, Yu Y. Deep contrast learning for salient object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016;478–487. doi:10.1109/CVPR.2016.56.
6. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC. SSD: Single shot multibox detector. In: Computer Vision—ECCV. Amsterdam, the Netherlands: Springer; 2016. p. 21–37.

doi:10.1007/978-3-319-46448-0\_2.

1. Oliver NM, Rosario B, Pentland AP. A Bayesian computer vision system for modeling human interactions. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2000;22(8):831–843. doi:10.1109/34.859891.
2. Oqubay M, Bottom L, Laptev I, Sivic J. Learning and transferring mid-level image representations using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014;1717–1724. doi:10.1109/CVPR.2014.224.
3. Ouyang W, Wang X. Joint deep learning for pedestrian detection. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). 2013;2056– 2063. doi:10.1109/ICCV.2013.267.
4. Monnet M, Paragios N, Ramesh V. Background modeling and subtraction of dynamic scenes. In: Proceedings of the 9th IEEE International Conference on Computer Vision. 2003;1305–1312. doi:10.1109/ICCV.2003.1238606.
5. Redmon J, Divola S, Kirchick R, Farhadi A. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016;779– 788. doi:10.1109/CVPR.2016.91.