International Journal of Engineering Research and Science & Technology

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 2 (2026)

ijerst.editor@gmail.com

editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 22, No. 2, 2026

Peer Reviewed, Referred & Indexed Journal

Darknet-Based YOLO Object Detection System with Web Integration for Real-Time Image Analysis

DADISETTI HEMAJYOTHI

PG Scholar. Department of MCA, DNR College, Bhimavaram, Andhra Pradesh

V.Sarala

(Assistant Professor), Master of Computer Applications, DNR College, Bhimavaram, Andhra Pradesh

ABSTRACT

Object detection is a critical task in computer vision that involves identifying and localizing objects within images or videos. With the advancement of deep learning, object detection has become more accurate and efficient, enabling its application in various domains such as surveillance, healthcare, autonomous driving, and smart cities. This project presents a Darknet-based YOLO object detection system integrated with a web-based interface for real-time image analysis.The system utilizes the YOLO (You Only Look Once) algorithm, implemented through the Darknet framework, which is known for its high speed and accuracy in object detection tasks. Unlike traditional detection methods that rely on multiple stages, YOLO performs detection in a single forward pass, making it highly efficient for real-time applications.

The proposed system allows users to upload images through a web interface built using Django. Once an image is uploaded, it is processed using the YOLO model to detect objects present in the image. The system identifies multiple objects, generates bounding boxes, and assigns confidence scores to each detection. The results are then displayed visually with annotated images, providing a clear understanding of detected objects.The system also stores detection results, including object labels and confidence scores, enabling users to review past detections. This feature enhances usability and supports data analysis. Additionally, the system includes user authentication and administrative functionalities to manage users and monitor system usage.

The Darknet framework is chosen for its optimized implementation of YOLO and its ability to leverage GPU acceleration for faster processing. The integration with Django ensures scalability, security, and ease of use.This project demonstrates the practical implementation of deep learning-based object detection in a web environment. It can be extended to support real-time video detection, IoT integration, and cloud deployment. The combination of Darknet and web technologies makes the system suitable for modern AI-driven applications.

Keywords: Object Detection, YOLO, Darknet, Deep Learning, Computer Vision, Image Processing, Neural Networks, Real-Time Detection, Django Web Application, AI-based Detection

I. INTRODUCTION

Computer vision has become a rapidly evolving field in artificial intelligence, with object detection being one of its most important applications. Object detection involves identifying objects in an image and determining their positions using bounding boxes. This technology is widely used in applications such as traffic monitoring, facial recognition, security surveillance, and industrial automation.Traditional object detection methods relied on handcrafted features and sliding window approaches, which were computationally expensive and inefficient. These methods struggled with complex scenes and variations in lighting, scale, and object orientation. With the advent of deep learning, particularly convolutional neural networks (CNNs), object detection has seen significant improvements in accuracy and efficiency.

YOLO (You Only Look Once) is one of the most popular deep learning algorithms for object detection. It processes the entire image in a single pass, making it significantly faster than region-based approaches like R-CNN. The Darknet framework provides an optimized implementation of YOLO, enabling real-time detection with high accuracy.This project aims to develop a web-based object detection system using the Darknet YOLO model integrated with the Django framework. The system allows users to upload images and receive detection results instantly. The use of a web interface makes the system accessible and user-friendly.The backend of the system handles image processing, model inference, and data storage. When an image is uploaded, the YOLO model detects objects and generates bounding boxes along with confidence scores. The results are stored and displayed to the user.The system also includes user authentication features, allowing users to register, log in, and manage their detection history. An admin dashboard provides insights into system usage and user activity.

The proposed system has wide-ranging applications in areas such as surveillance, traffic management, retail analytics, and healthcare. It demonstrates how deep learning models can be effectively deployed in real-world applications using web technologies.

II. LITERATURE SURVEY (WITH EXISTING METHODS)

Object detection has undergone significant advancements over the past decade, transitioning from traditional computer vision techniques to deep learning-based methods. Early approaches such as Haar cascades and Histogram of Oriented Gradients (HOG) relied on handcrafted features and were limited in their ability to handle complex scenarios.The introduction of deep learning revolutionized object detection. Region-based Convolutional Neural Networks (R-CNN) marked a major milestone by using CNNs for feature extraction and classification. Variants such as Fast R-CNN and Faster

R-CNN improved performance by optimizing region proposal mechanisms. However, these methods were computationally expensive and unsuitable for real-time applications.

Single-stage detectors such as SSD (Single Shot MultiBox Detector) and YOLO addressed these limitations by performing detection in a single pass. YOLO divides the image into grids and predicts bounding boxes and class probabilities simultaneously. This approach significantly improves speed while maintaining competitive accuracy.The Darknet framework, developed for YOLO, provides an efficient implementation optimized for performance. YOLO has evolved through multiple versions, each improving accuracy and speed. YOLOv3 and later versions introduced better feature extraction and multi-scale detection capabilities.

Recent research has focused on improving detection accuracy for small objects, handling occlusions, and enhancing performance in low-light conditions. Techniques such as attention mechanisms, feature pyramids, and transformer-based models have been proposed.In addition, web-based deployment of object detection systems has gained popularity. Frameworks like Django and Flask are used to build user interfaces for AI applications, enabling real-time interaction and accessibility.

The proposed system leverages the Darknet YOLO model and integrates it with a Django web application. It combines the strengths of deep learning and web technologies to create an efficient and scalable object detection system.

III. EXISTING SYSTEM

Existing object detection systems often rely on traditional methods or earlier deep learning models, which have several limitations. Traditional approaches use handcrafted features and sliding window techniques, resulting in high computational cost and low accuracy. These systems are not suitable for real-time applications and struggle with complex scenes.Early deep learning models such as R-CNN and Fast R-CNN improved detection accuracy but required multiple stages for processing. This increased computational complexity and reduced speed, making them impractical for real-time use.

Some modern systems use single-stage detectors like SSD or earlier versions of YOLO. While these models offer better speed, they may not achieve the same level of accuracy as newer YOLO implementations in Darknet.Another limitation of existing systems is the lack of user-friendly interfaces. Many systems operate as standalone applications without web integration, limiting accessibility and scalability. They often do not include features such as user authentication, detection history, and administrative control.Additionally, existing systems may not provide detailed insights such as confidence scores, object counts, or descriptive summaries. Visualization of results is also limited, making it difficult for users to interpret detection outputs.The proposed system addresses these limitations by using the Darknet YOLO model for high-speed and accurate detection, combined with a Django-based web interface for improved usability, scalability, and functionality.

IV. PROPOSED METHOD

The proposed system is a web-based object detection platform that integrates the YOLO (You Only Look Once) deep learning model implemented using the Darknet framework with a Django-based web application. The system is designed to provide fast, accurate, and real-time object detection through an intuitive user interface.The system allows users to upload images via a web portal. Once an image is uploaded, it is processed using the YOLO model, which detects objects in a single forward pass. YOLO is chosen because it achieves an optimal balance between detection accuracy and computational efficiency, making it suitable for real-time applications . The system generates bounding boxes, class labels, and confidence scores for each detected object.

Detection results are stored in a database, including object names, counts, confidence scores, and annotated images. A descriptive summary is automatically generated to improve interpretability. Users can access their detection history, enabling tracking and analysis of previous results.An admin module is incorporated to monitor system activity, including user statistics and detection records. The system ensures secure access through authentication and role-based authorization.The proposed system is scalable and can be extended to support real-time video detection, cloud-based deployment, and IoT integration. It addresses limitations of traditional systems by combining high-speed deep learning models with a web-based platform for accessibility and usability.

V. IMPLEMENTATION

The implementation of the proposed system is carried out using Python, Django, OpenCV, and the YOLO model implemented via the Darknet or Ultralytics framework. The system is divided into several modules, including user authentication, image processing, object detection, and data management.The frontend is developed using HTML, CSS, and Django templates. It provides an interface for user registration, login, image upload, and result visualization. Django's built-in authentication system is used to manage users securely, while additional user details are stored in a separate profile model.

When a user uploads an image, it is stored in the media directory, and a record is created in the database. The backend processes the image using OpenCV and feeds it into the YOLO model for detection. YOLO processes the image in a single pass, making it significantly faster than traditional multi-stage detectors .The detection results include bounding boxes, object labels, and confidence scores. These outputs are extracted and stored in JSON format. An annotated image is generated by overlaying bounding boxes on the original image and saved for visualization.

The system also generates a textual description of the detected objects. This includes object counts, unique object types, and average confidence scores. This feature enhances the interpretability of the results.

Error handling mechanisms are implemented to ensure system reliability. If the YOLO library is not installed or fails during execution, appropriate error messages are displayed

to the user.The prediction results are displayed on a separate page, where users can view annotated images, object details, and confidence scores. The system also maintains a history of all detections, allowing users to review previous results.The admin dashboard provides insights into system usage, including total users and detection counts. It enables administrators to monitor activity and manage users effectively.Overall, the implementation ensures efficiency, scalability, and user-friendliness while leveraging deep learning for accurate object detection.

VI. ALGORITHMS

The system uses the YOLO (You Only Look Once) algorithm as its primary object detection technique.

YOLO Algorithm

YOLO is a single-stage object detection algorithm that processes the entire image in one pass. Unlike region-based methods, YOLO divides the image into a grid and predicts bounding boxes and class probabilities simultaneously. This approach significantly improves detection speed and efficiency.

YOLO uses a convolutional neural network (CNN) backbone for feature extraction. It then applies detection layers to predict object locations and classes. The latest versions use anchor-free detection mechanisms and improved feature pyramid networks for better accuracy.

Convolutional Neural Networks (CNN)

CNNs are used for extracting spatial features from images. They consist of convolutional layers, pooling layers, and fully connected layers, enabling the model to learn complex patterns in image data.

Non-Maximum Suppression (NMS)

After detection, multiple overlapping bounding boxes may be generated. NMS is applied to remove redundant boxes and retain the most accurate detections.

Recent research highlights that YOLO-based models achieve high accuracy and real-time performance, making them suitable for applications such as surveillance and autonomous systems .

VII. SYSTEM DESIGN

The system follows a three-tier architecture consisting of presentation layer, application layer, and data layer.

1. Presentation Layer

This layer includes the user interface developed using Django templates. It allows users to:

- Register and log in
- Upload images
- View detection results
- Access detection history

The interface is designed to be responsive and user-friendly.

2. Application Layer

The application layer is the core of the system. It handles:

- User authentication
- Image upload and storage
- YOLO model execution
- Result processing

Key modules include:

- **Authentication Module**: Handles login, signup, logout
- **Detection Module**: Processes images using YOLO
- **History Module**: Stores detection results
- **Admin Module**: Provides analytics and monitoring

YOLOv8 and its variants use advanced architectures such as feature pyramid networks and anchor-free detection, improving performance across diverse datasets .

3. Data Layer

The data layer includes:

- Database (SQLite/MySQL) for storing user and detection data
- Media storage for images
- JSON storage for detection results

Workflow

1. User logs into the system
2. Uploads an image
3. Image is stored on the server
4. YOLO processes the image

5. Objects are detected
6. Results are stored
7. Output is displayed

Design Features

- Modular and scalable architecture
- Secure authentication system
- Efficient storage and retrieval
- Real-time detection capability

SYSTEM DESIGN IMAGES

Now-a-days increase usage of networks open new ways for attackers to hack or send malicious request to user system. Darknet also is one of the type attack and all existing Machine Learning algorithms were trained on single Darknet2020 CIC dataset but attackers may find new request variable to deceive existing algorithms. To overcome from this issue author of this paper merging data from two different datasets such as Darknet2020 and ISCX to detect all possible change variables attackers may use.

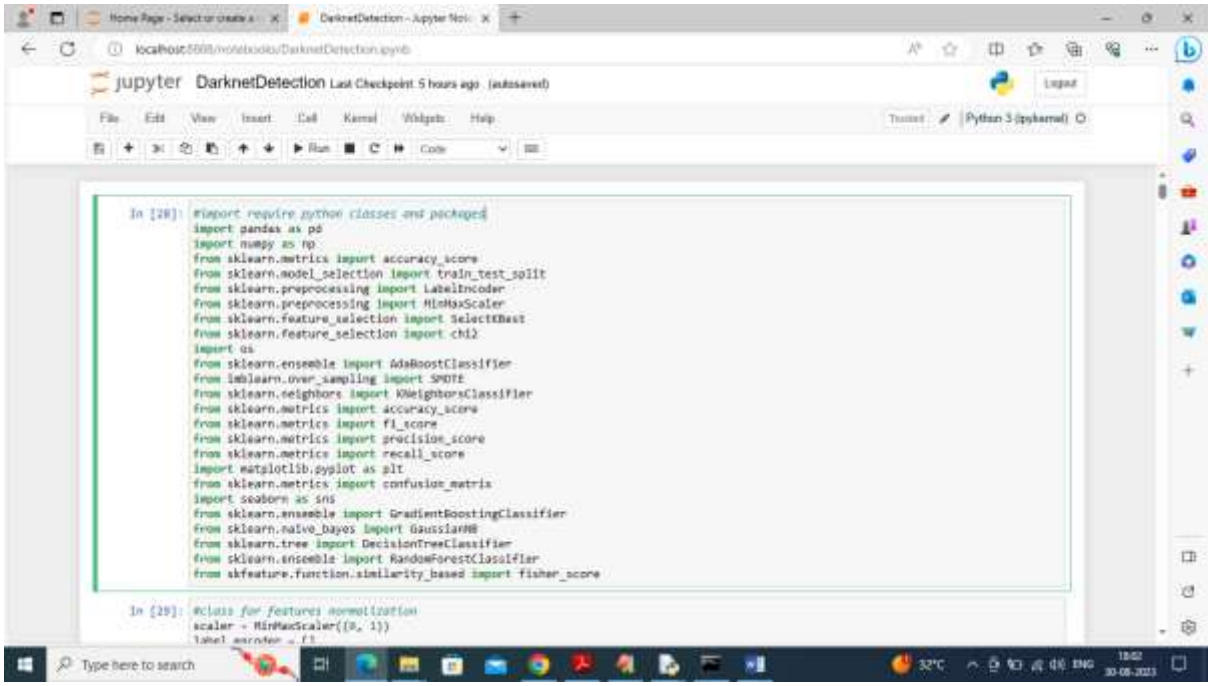
In propose work both datasets will be input to Fisher algorithm to select high ranking features and then train different algorithms by using different class labels listed below

1) DTC1: First entire dataset will be divided into Darknet and Benign classes where TOR and VPN labels will be consider as Darknet and NONTOR and NONVPN will be consider as Benign to form a binary class labels. Selected features and binary class labels will get trained with Decision Tree, KNN, Naïve Bayes and ADABOOST
2) DTC2 Multi classifier: DTC1 class labels will be divided into 4 different class labels such as TOR, NONTOR, VPN and NONVPN and then trained with Random Forest, Decision Tree, Naïve Bayes, KNN and Gradient Boosting
3) DTC3 Multi classifier: here entire dataset will be split into 8 different traffic categories and trained with 5 different algorithms such as Random Forest, Decision Tree, Naïve Bayes, KNN and Gradient Boosting

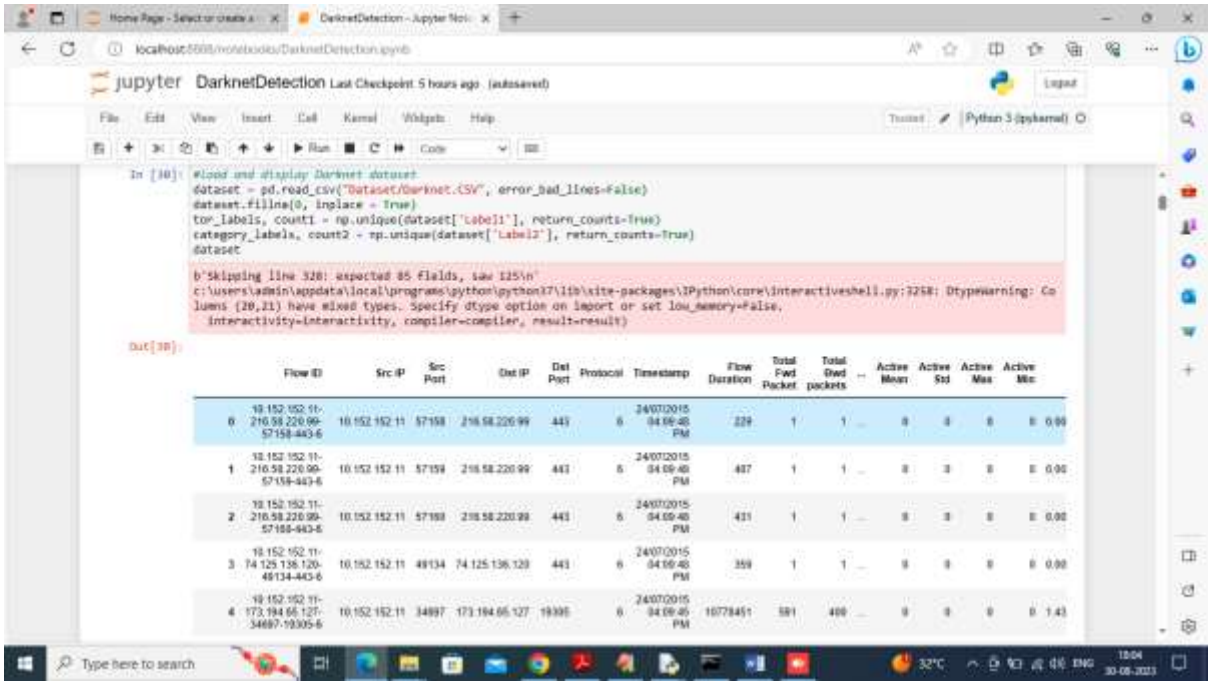
Propose algorithm training on two different datasets able to get high accuracy compare to existing single dataset trained algorithms.

SCREEN SHOTS

We have coded this project using JUPYTER NOTEBOOK and below are the code and output screens with blue color comments

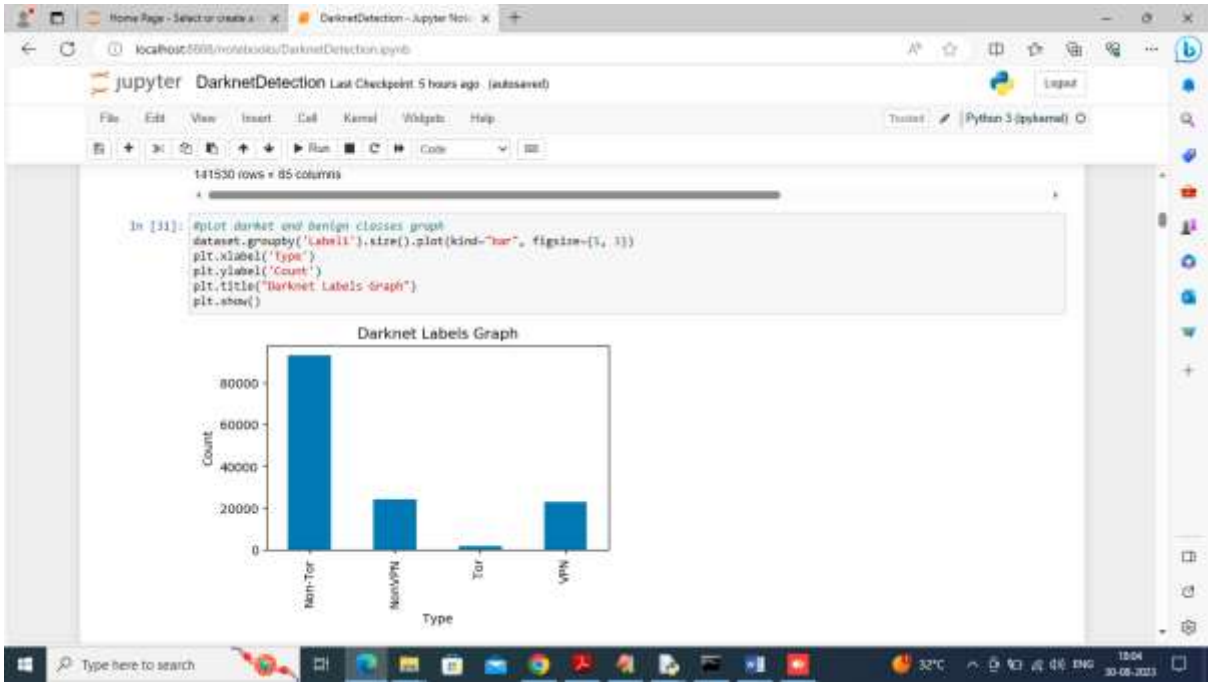


In above screen importing require python classes and packages

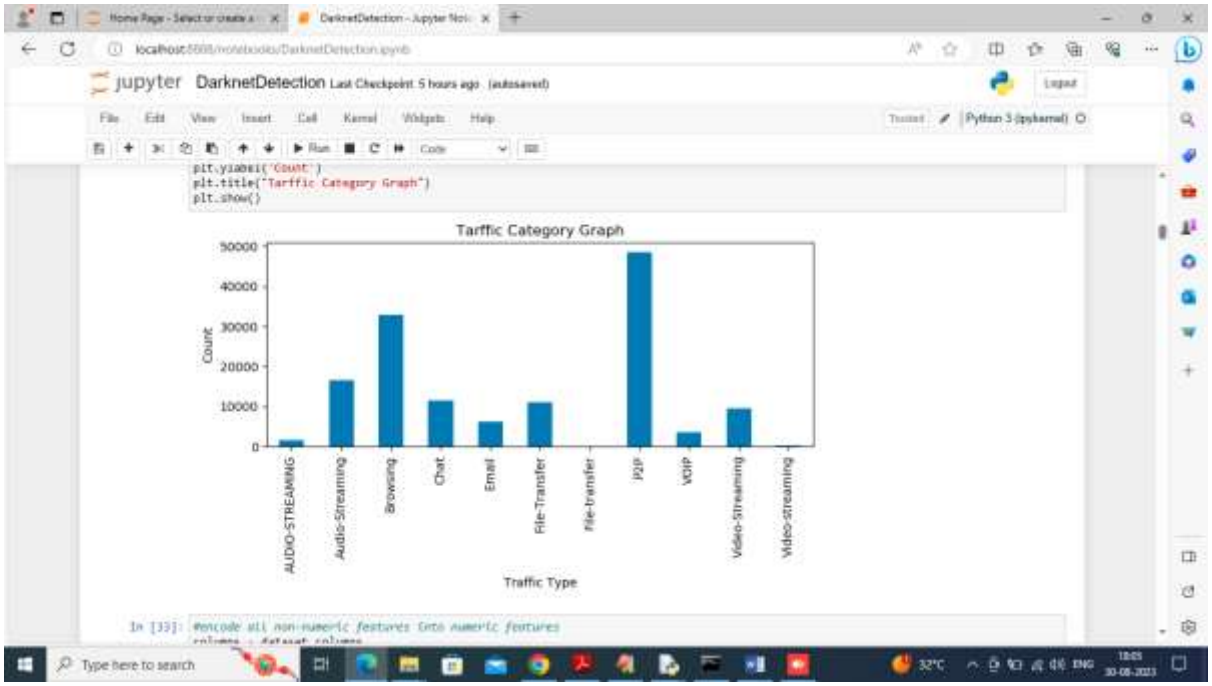


In above screen loading and displaying Dartket2020 dataset to training existing algorithms

International Journal of Engineering Research and Science & Technology

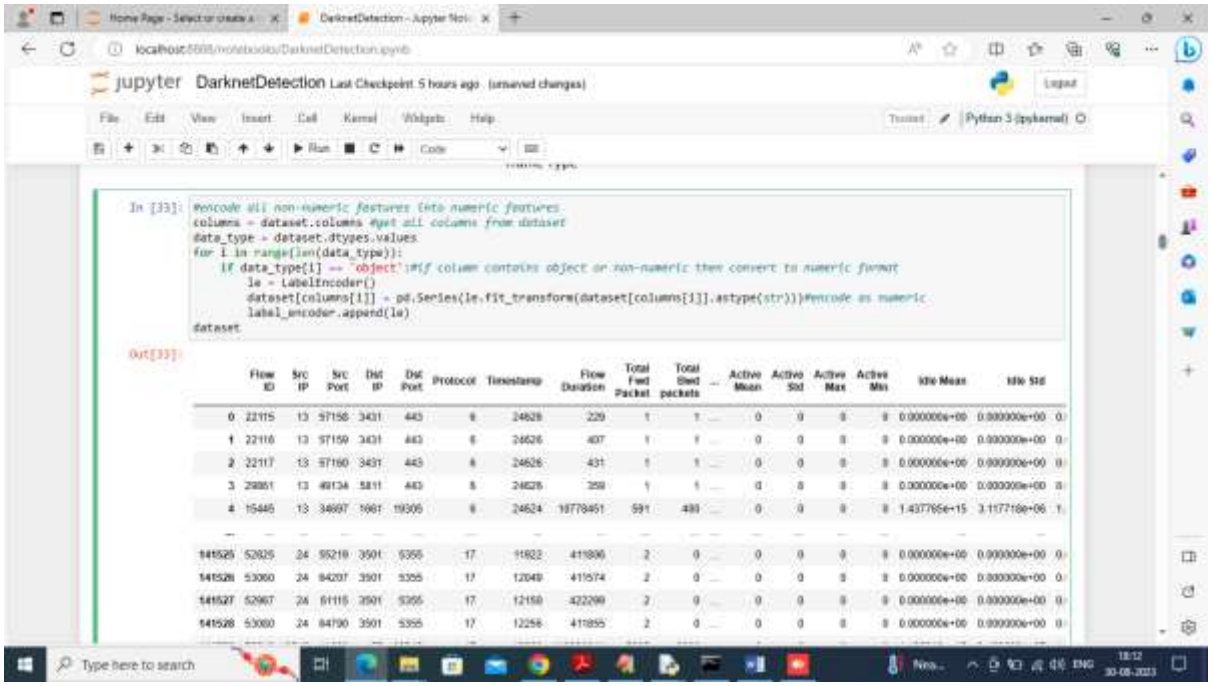


In above screen plotting graph of different Darknet and benign classes graph where x-axis represents names and y-axis represents count

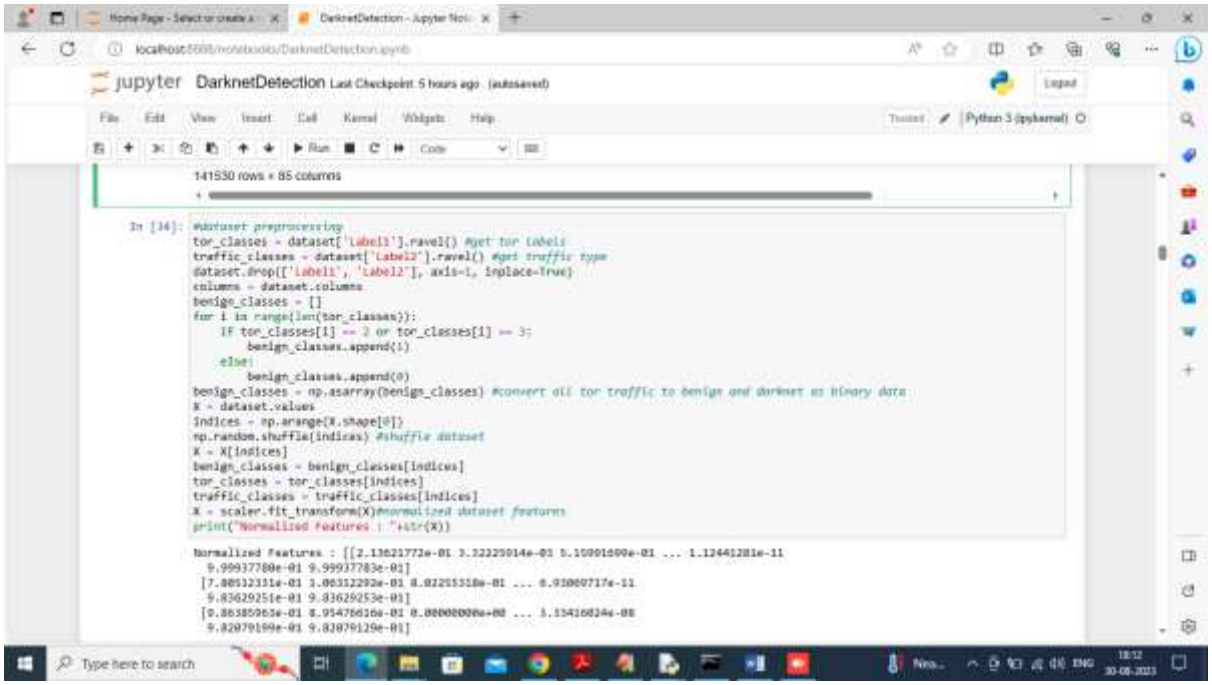


In above screen plotting graph of different traffic category found in dataset where x-axis represents traffic category and y-axis represents count

International Journal of Engineering Research and Science & Technology

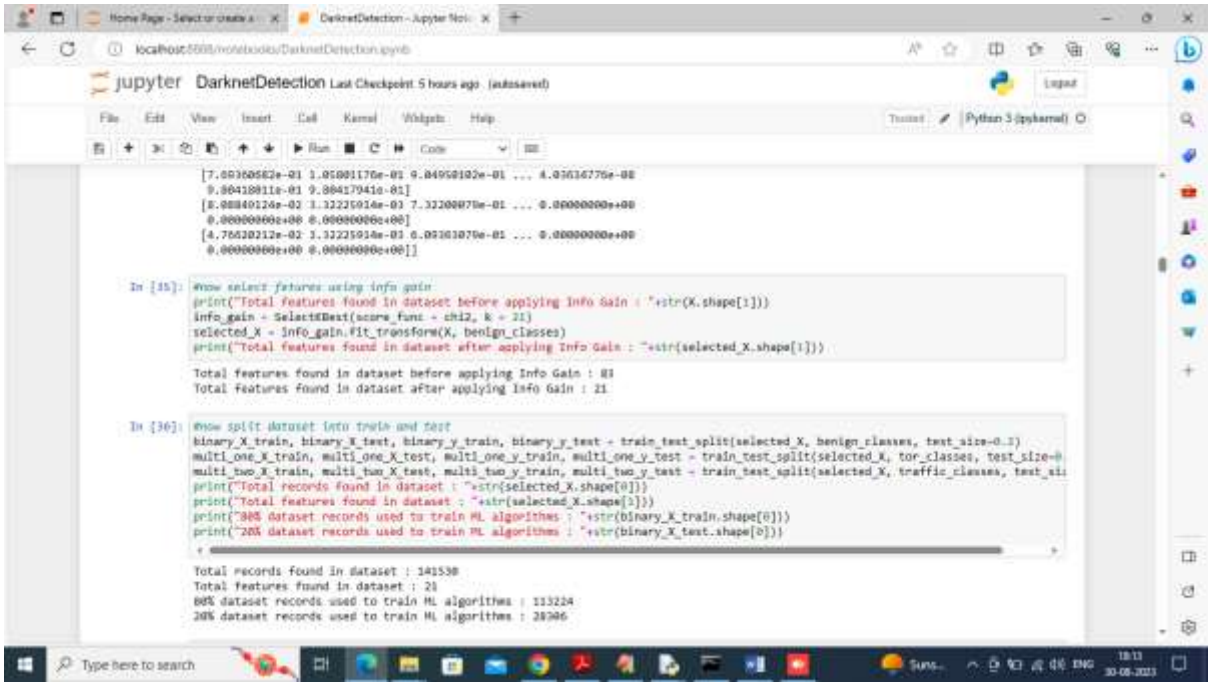


In above screen converting all non-numeric columns to numeric columns

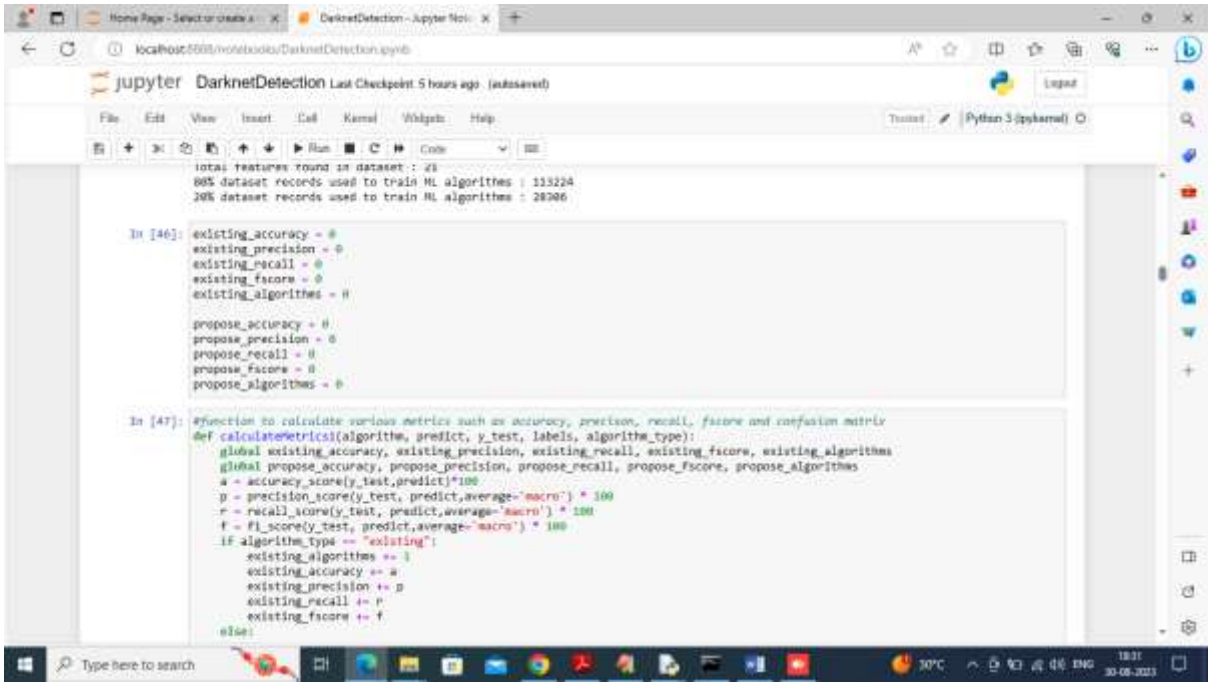


In above screen applying pre-processing techniques such as extracting X and y labels and then shuffling and normalizing dataset values

International Journal of Engineering Research and Science & Technology

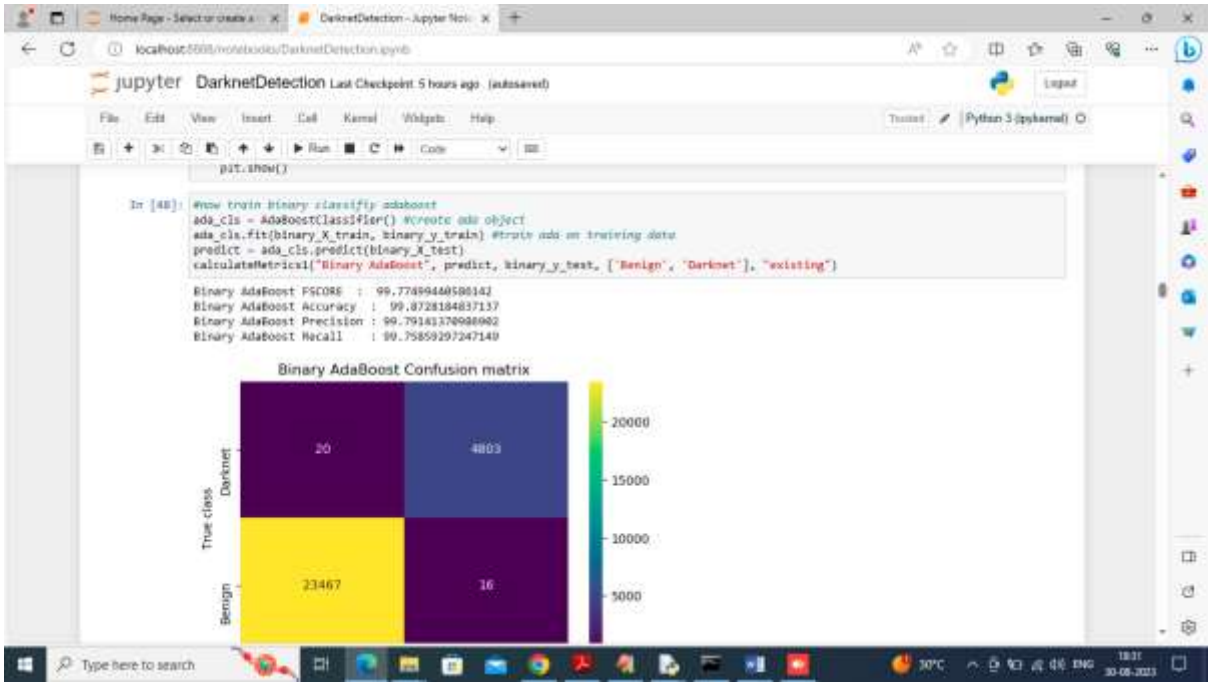


In above screen for existing technique applying Information Gain to select relevant features and then selected 21 best features and then splitting dataset into train and test where application using 80% dataset for training and 20% for testing

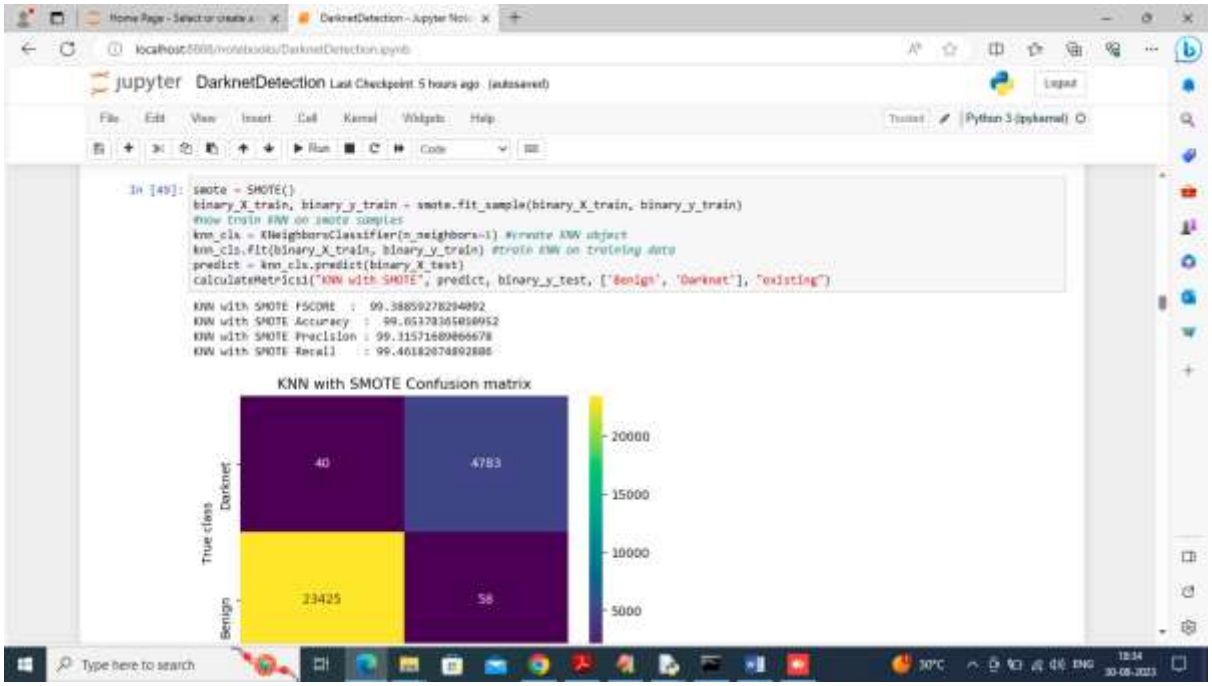


In above screen defining function to calculate accuracy and other metrics for both existing algorithms and propose algorithms

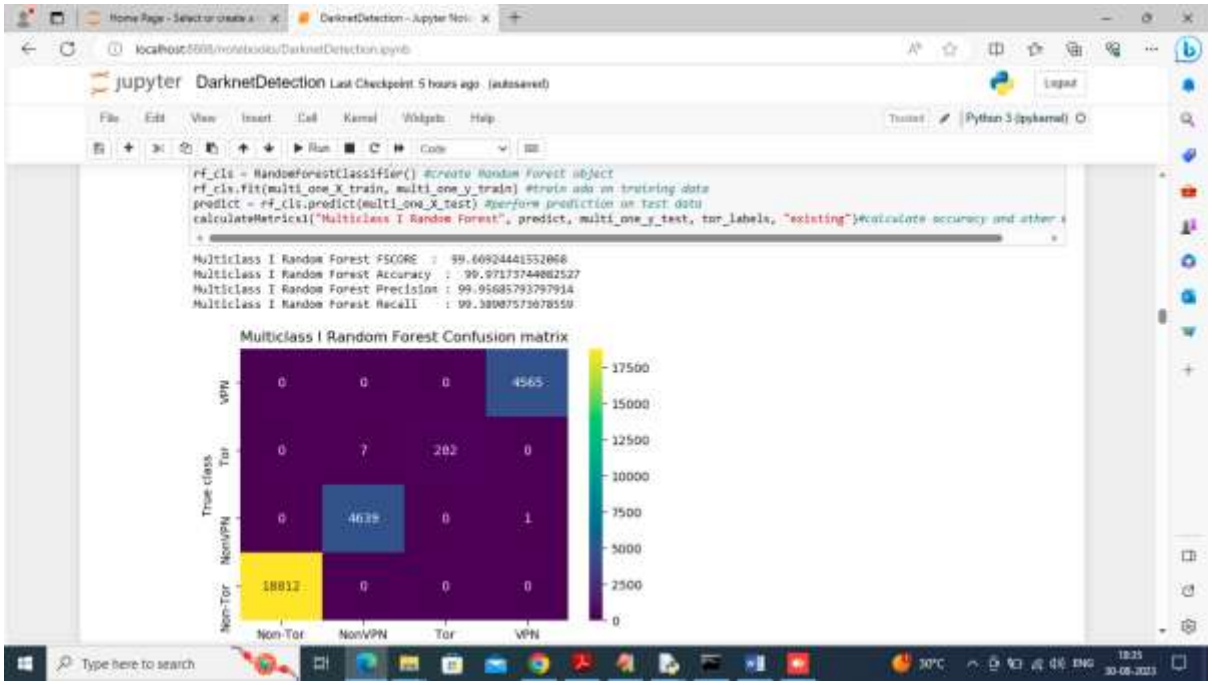
International Journal of Engineering Research and Science & Technology



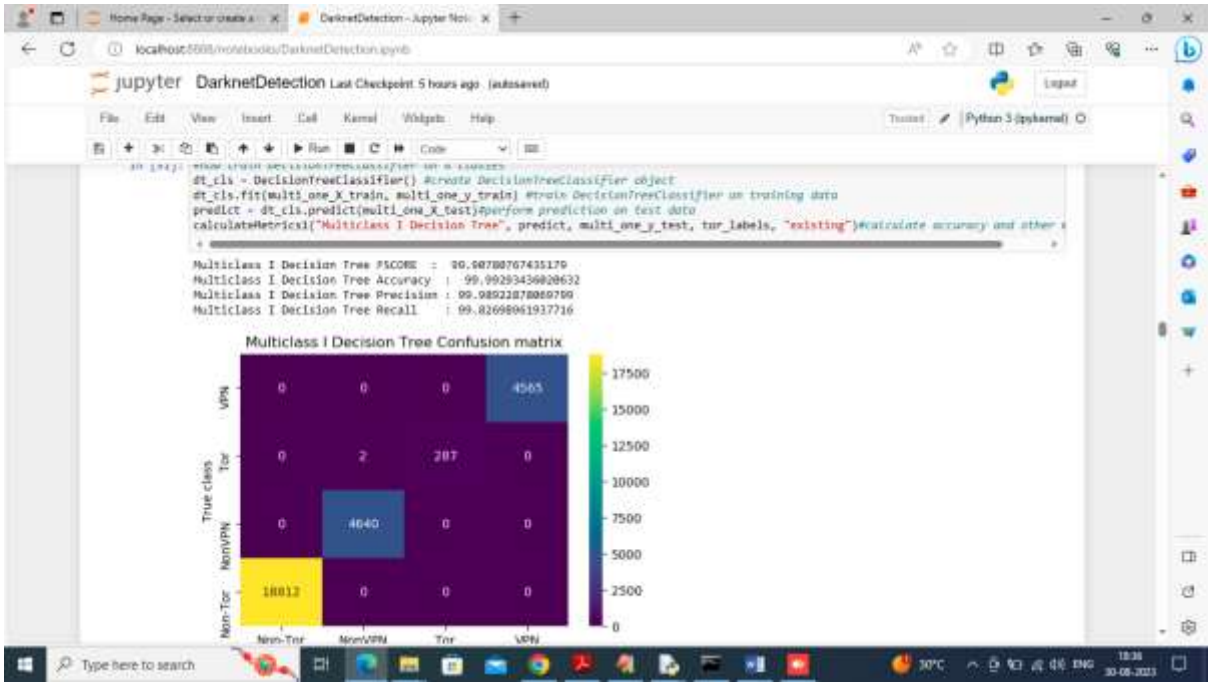
In above screen training ADABOOST on binary classes using existing single dataset and then ADABOOST got 99.77% accuracy and can see other metrics also and in confusion matrix graph x-axis represents Predicted Labels and y-axis represents true labels and both blue color boxes contains incorrect prediction count which are very few and different color boxes like yellow and other color box contains correct prediction count



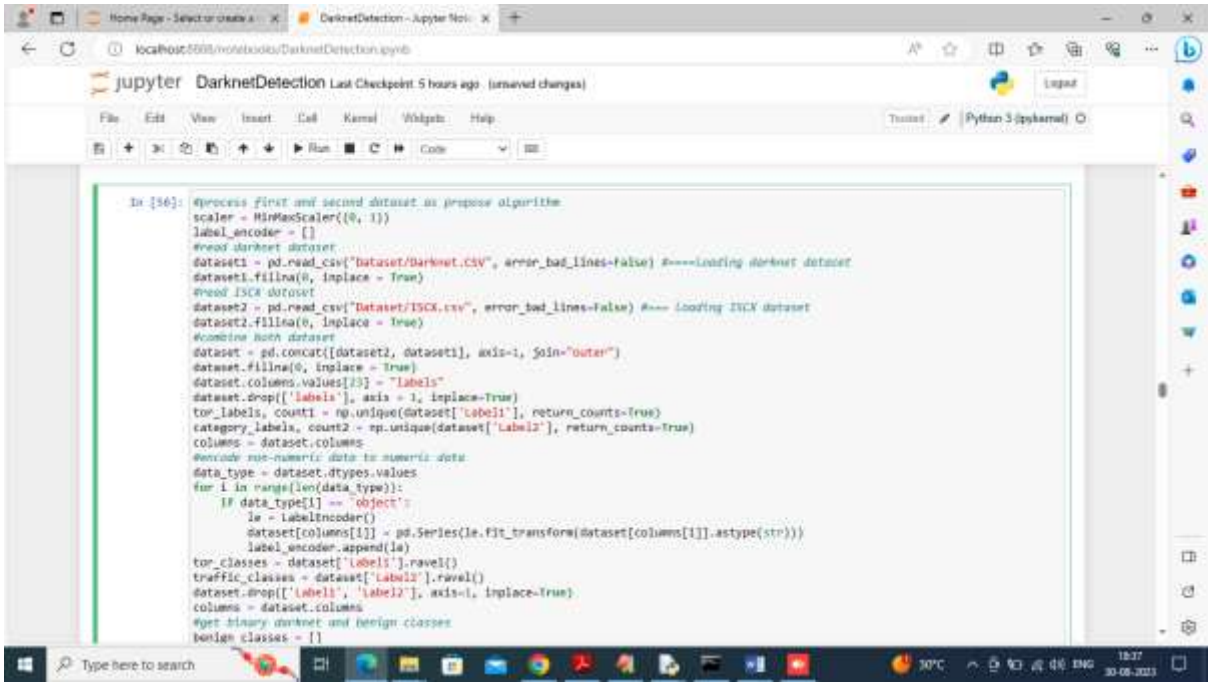
In above screen training existing KNN wit SMOTE and got accuracy as 99.18% and can see other metrics also



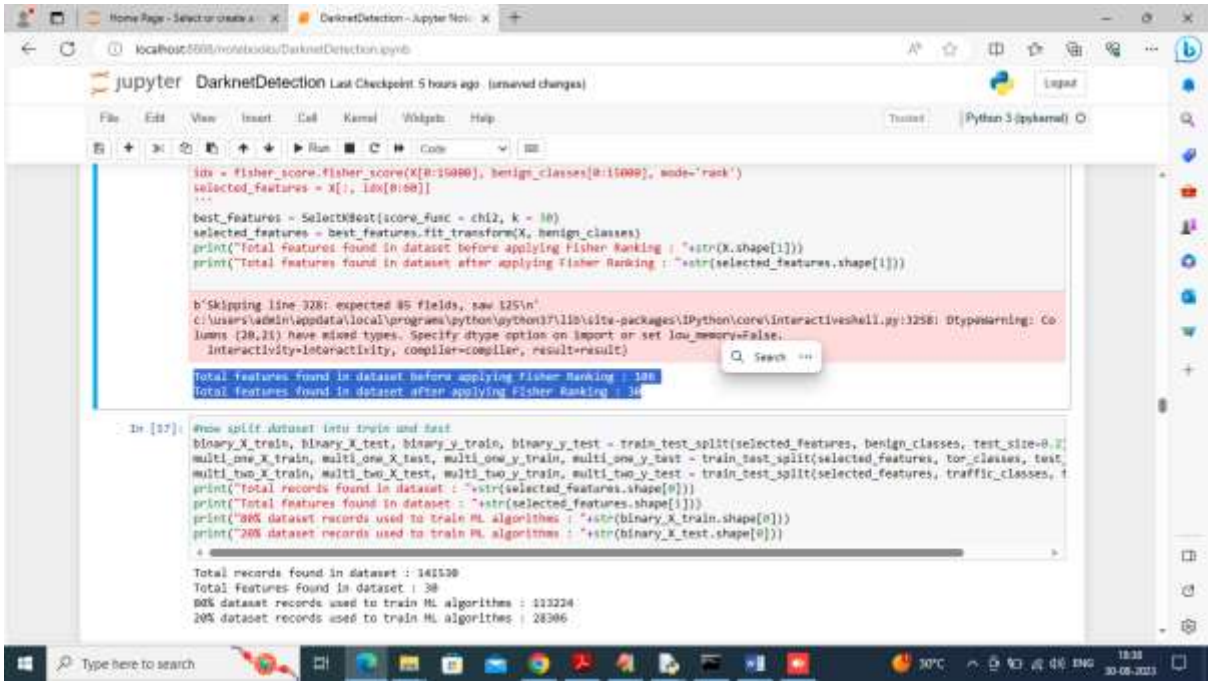
In above screen training existing Random Forest on 4 classes as Multi class I dataset and got accuracy as 99.66%



In above screen training decision tree and similarly we trained all existing algorithms on single existing dataset and now in below screen showing propose algorithms by combining two different datasets

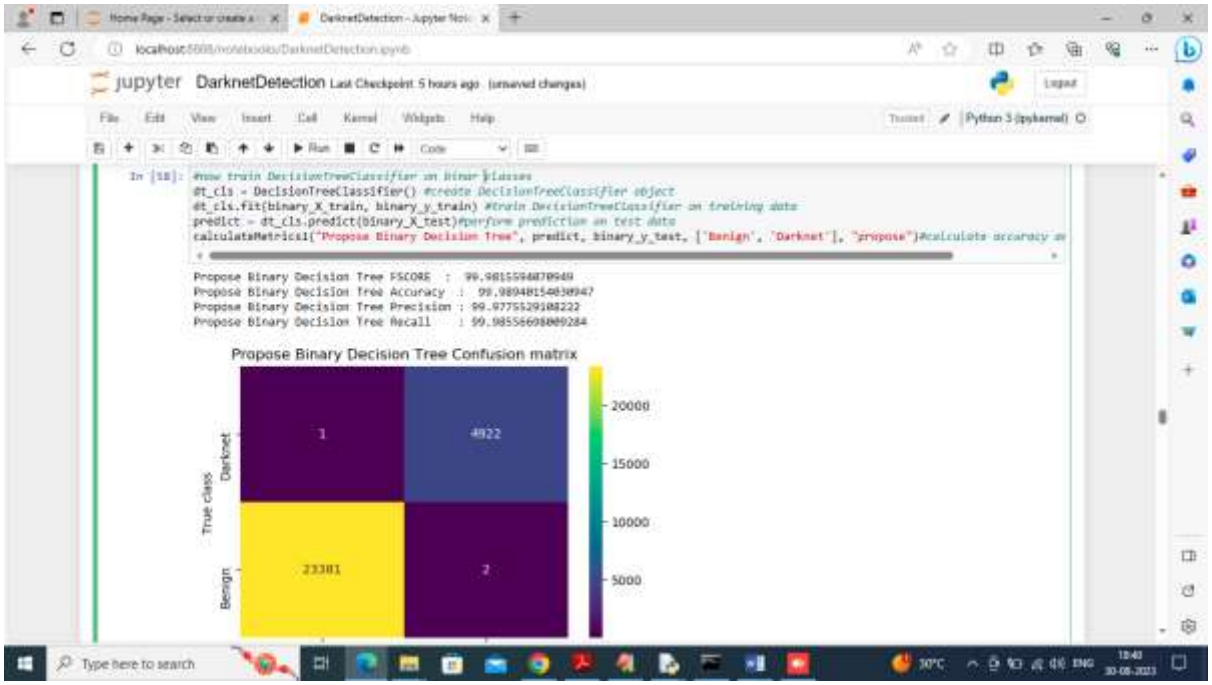


In above screen for propose work we are loading both Darknet2020 and ISCX dataset and then applying preprocessing techniques

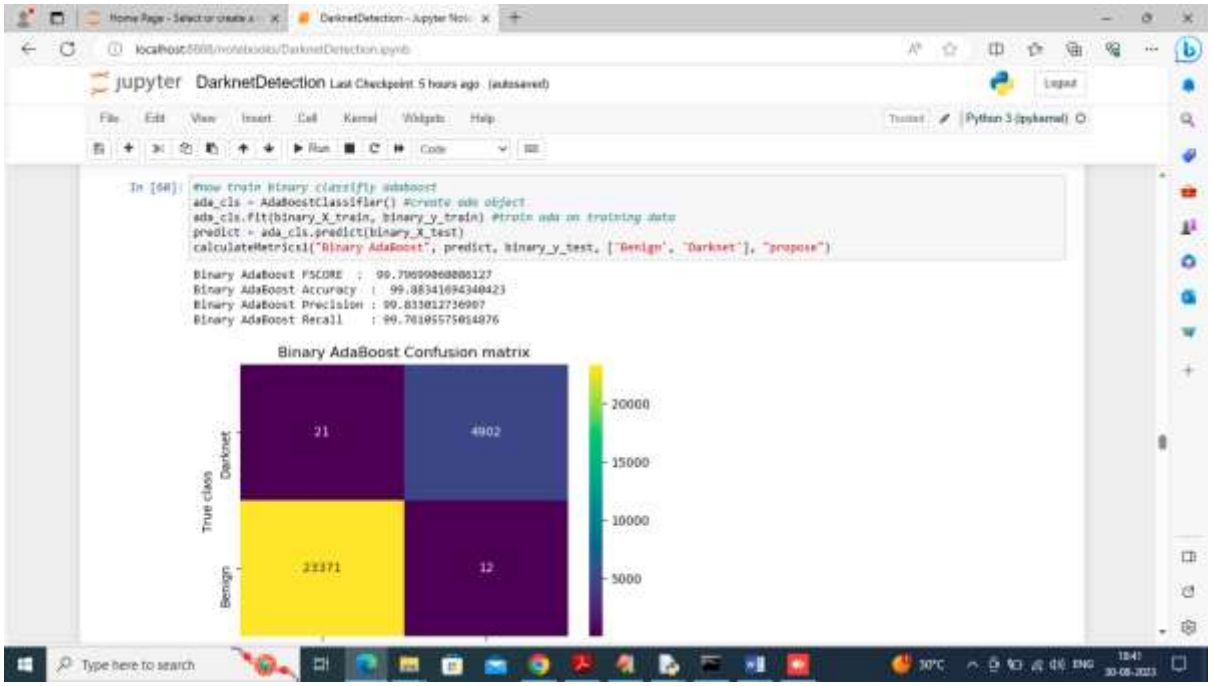


International Journal of Engineering Research and Science & Technology

In above screen from propose dataset applying FISHER algorithm to select 30 best features and then splitting into train and test

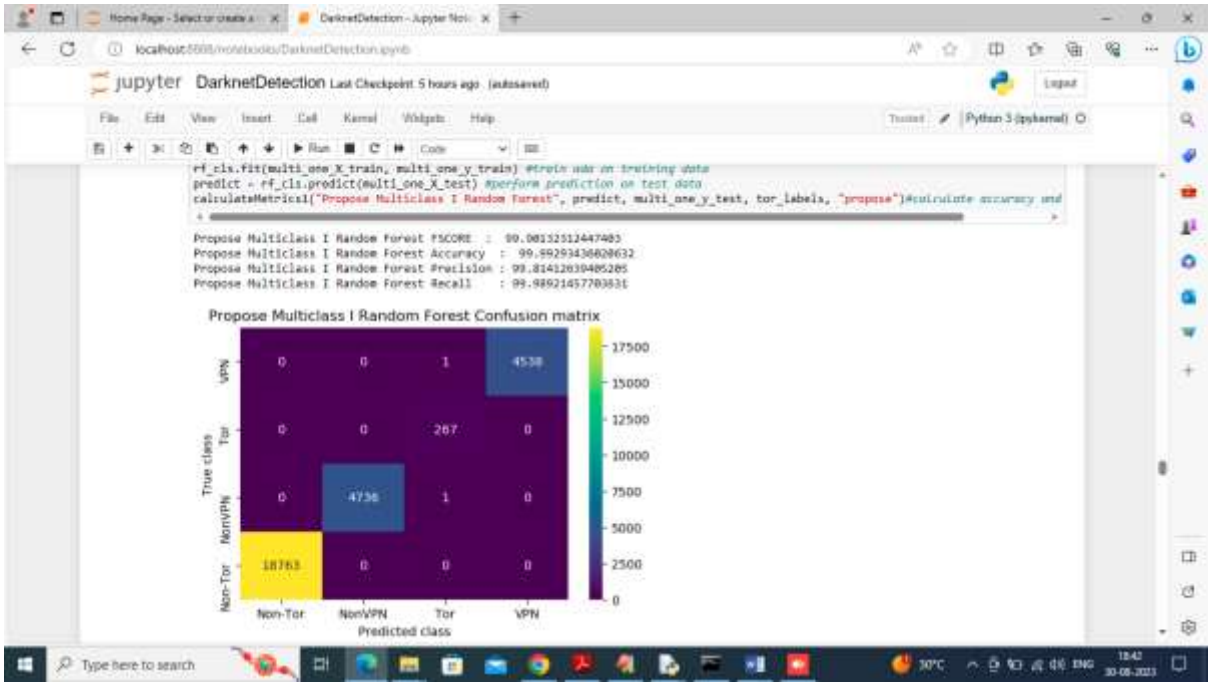


In above screen training propose Decision tree on both dataset features and got accuracy as 99.98% and can see other metrics also

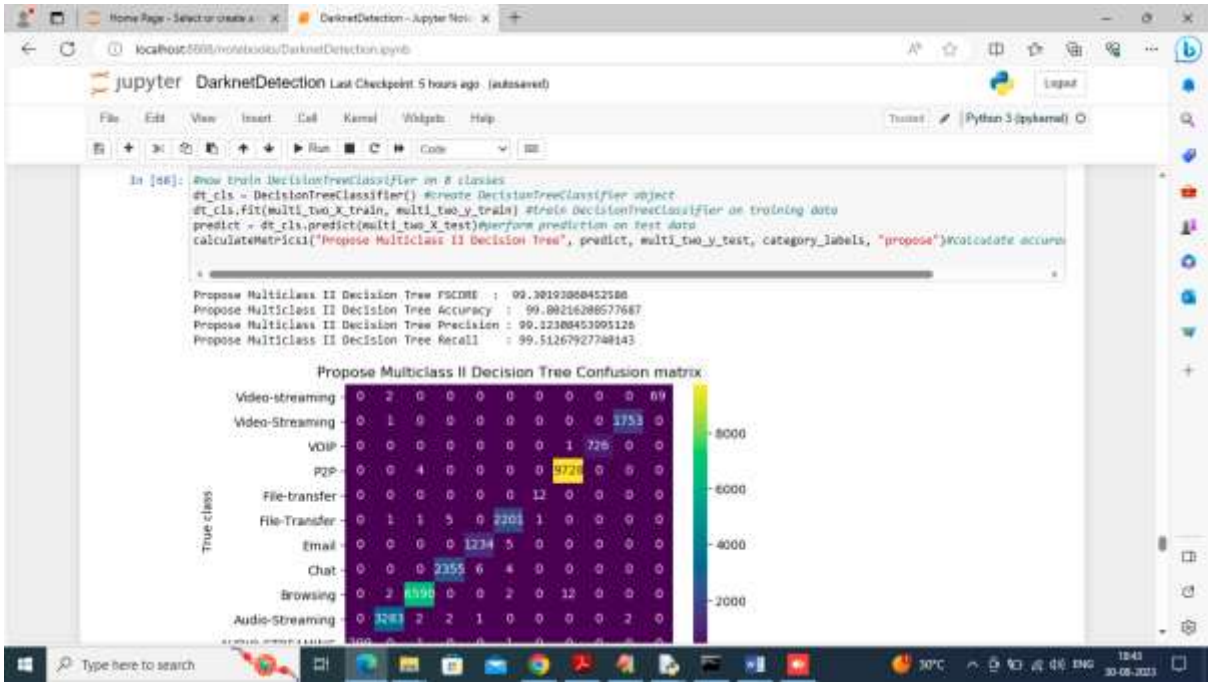


In above screen training propose ADABOOST on binary classes and got accuracy as 99.79%

International Journal of Engineering Research and Science & Technology

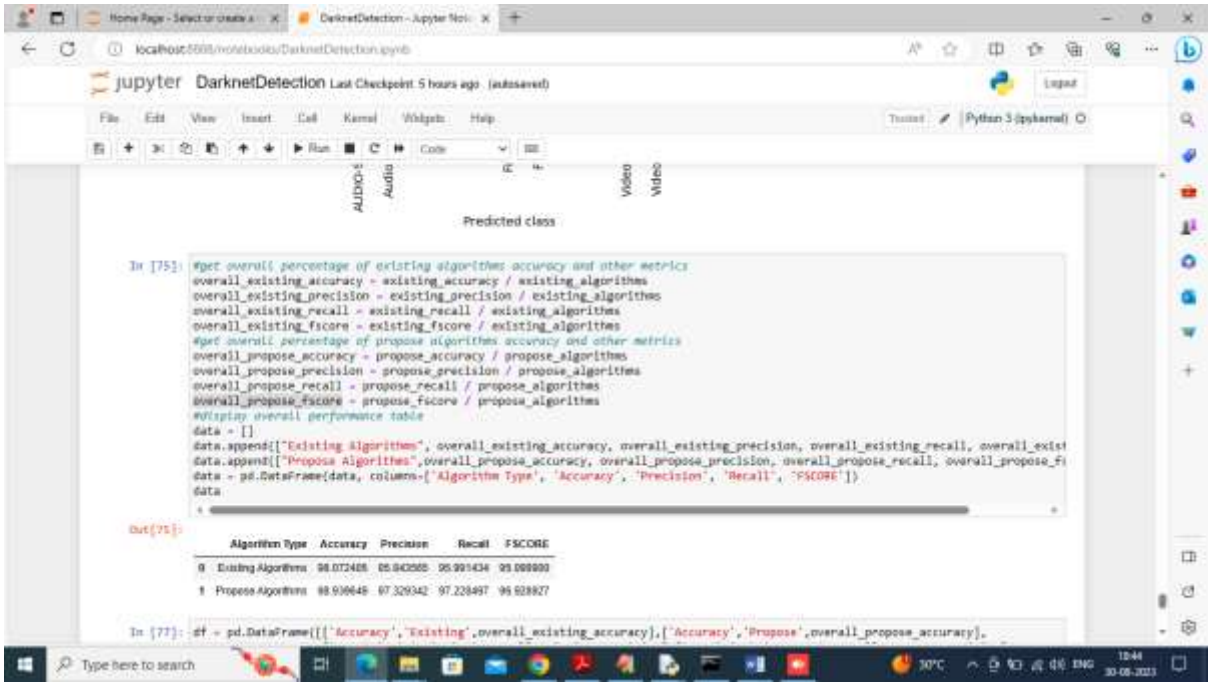


In above screen training Random Forest on propose dataset with 4 classes as TOR, NONTOR, VPN and NONVPN and got accuracy as 99.90%

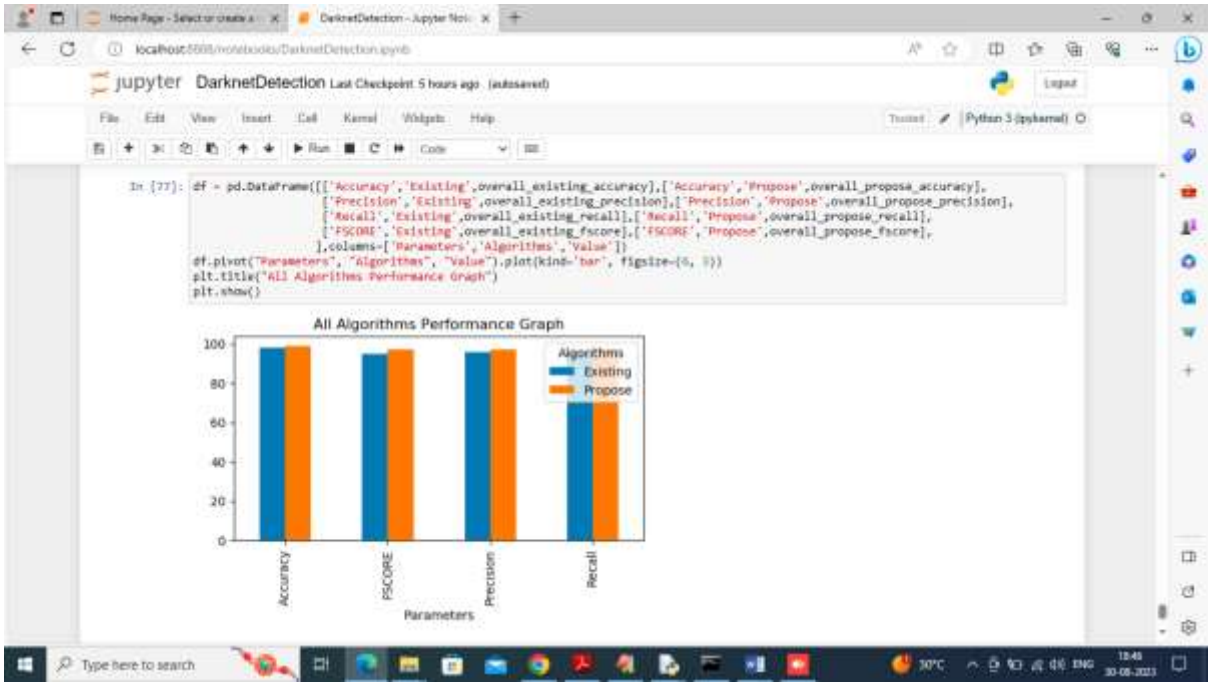


In above screen training Decision Tree 8 different classes as MULTICLASS II and got accuracy as 99.30% and can see other metrics also. Similarly we trained all algorithms from propose and existing LIST. Below is the overall accuracy of all algorithms on existing and propose work

International Journal of Engineering Research and Science & Technology



In above screen in tabular format we can see overall accuracy and other metrics from both propose and existing algorithms and in both PROPOSE algorithms got high accuracy



In above screen displaying existing and propose performance in graph format where x-axis represents metrics and blue bar represents existing metrics and orange bar represents

propose metrics and from above results we can say combining and selecting features from multiple datasets can increase ML performance.

VIII. CONCLUSION

This project presents a Darknet-based YOLO object detection system integrated with a Django web application. The system effectively combines deep learning and web technologies to provide accurate and real-time object detection.The use of YOLO enables fast processing and high accuracy, making the system suitable for real-world applications such as surveillance, traffic monitoring, and industrial automation. The integration with Django enhances usability, scalability, and security.The system includes features such as user authentication, detection history, and admin dashboard, which improve user experience and system management. The ability to store and analyze detection results adds value for users and administrators.

Despite its advantages, the system may face challenges such as dependency on computational resources and model availability. However, these can be addressed through optimization and hardware acceleration.Future enhancements may include real-time video processing, cloud deployment, and integration with IoT devices. Advanced models and techniques can also be incorporated to improve detection accuracy in challenging conditions.Overall, the project demonstrates the effectiveness of YOLO-based object detection and highlights the potential of integrating AI with web technologies for practical applications.

REFERENCES

1. Silva et al., “Recurrent YOLOv8 Framework,” 2025
2. Ali & Zhang, “YOLO Framework Review,” 2024
3. Springer, “YOLOv8 for Behavioral Research,” 2024
4. Moussaoui et al., “YOLOv8 for Vehicle Detection,” 2024
5. Jahan et al., “Enhanced YOLOv8 for Safety,” 2025
6. Varghese et al., “YOLOv8 Performance Study,” 2024
7. MDPI Sensors, “SRE-YOLOv8 Model,” 2024
8. ScienceDirect, “Small Object Detection using YOLOv8,” 2024
9. ADICS Conference YOLOv8 Study, 2024
10. ArXiv, “YOLOv8 Architecture Analysis,” 2024
11. ArXiv, “SOD-YOLOv8 Small Object Detection,” 2024
12. ArXiv, “YOLOv8-AM Attention Model,” 2024
13. ArXiv, “YOLOv8 vs Transformer Models,” 2024
14. Scientific Reports, “YOLOv8 Enhancements,” 2025
15. MDPI Computers, “YOLO Evolution Study,” 2024