https://doi.org/10.63356.stes.ing.2025.002 УДК 528.8.044:004.42`275

USER-FRIENDLY INTERFACE FOR TRAINING AND DEPLOYMENT OF YOLO MODELS FOR OBJECT DETECTION

Authors: MARKO SIMONIČ, Goran Munđar, Rudolf Leon Filip
Email: marko.simonic@um.si
Mentors: Assoc. Prof. Simon Klančnik, Assoc. Prof. Uroš Župerl
Faculty of Mechanical Engineering, University of Maribor

Introduction: The growing demand for efficient real-time object detection algorithms has led to the development of the YOLO (You Only Look Once) architecture, which has revolutionized the field of machine vision. This research introduces a user-friendly interface designed for students and researchers to easily train, validate, and deploy YOLO models.

Aim: The aim of this research is to develop a user-friendly software interface that enables data import, hyperparameter configuration, model training, and object detection on images, on videos, and in real-time via webcam, with a focus on educational and research applications.

Materials and Methods: The software was developed using Python with Ultralytics YOLO, OpenCV, and PyQt5 libraries for the user interface. It consists of four modules: a training module, an image detection module, a video detection module, and a real-time detection module. Development included testing on Windows platforms with NVIDIA GPU support, using public datasets like COCO for validation.

Results: The developed interface achieved high efficiency, with detection speeds up to 30 FPS in real-time. The modules were tested on various datasets, showing precision up to 95% on validation sets. The user interface enables intuitive navigation, and implementation on the recommended hardware configuration ensures reliable performance.

Conclusion: This research demonstrates the potential of the user-friendly interface to facilitate work with YOLO models in educational and research environments. By automating processes, the system reduces barriers for users and improves access to advanced machine vision techniques. Future work will focus on integrating newer YOLO variants and support for multiple operating systems.

Keywords: YOLO; object detection; machine vision; user-friendly interface; real-time detection

INTRODUCTION

Object detection has become a foundational capability in modern manufacturing, robotics, and research, serving as a key enabler for automation, inspection, and intelligent decision-making [1], [2]. However, training and deploying high-quality models still demands a non-trivial mix of data preparation, environment setup, and framework know-how. Among existing algorithms, the YOLO (You Only Look Once) family has gained wide popularity due to its balance between accuracy and real-time performance [3], [4]. The evolution of YOLO—from the original version to the latest YOLOv11—has introduced significant improvements in network depth, feature fusion, and computational efficiency [5]–[7]. Despite these advances, students and practitioners often face steep entry barriers: formatting datasets, tuning hyperparameters, wiring together scripts for image, video, and

webcam inference, and organizing outputs for analysis [8].

This paper presents DeepDetect, a user-friendly graphical interface that streamlines the entire workflow for YOLO-based object detection without automating it behind the scenes. The contribution is a guided, integrated environment that helps users (i) import custom datasets in standard YOLO format, (ii) configure and launch training with transparent control of key hyperparameters, (iii) run inference on images, videos, and live webcam feeds, and (iv) save results (annotated media, CSV/Excel/JSON logs, confusion matrices, and performance curves such as Precision/Recall and F1) for reproducible analysis. The interface supports multiple YOLO generations (e.g., YOLOv8–YOLOv11) and can leverage GPU acceleration (CUDA) when available, while remaining usable on CPU-only systems for teaching and quick experiments.

Designed primarily for education and research, DeepDetect reduces the operational friction of working with YOLO by replacing scattered command-line steps with a cohesive UI built in Python using the Ultralytics YOLO framework, OpenCV, and CustomTkinter [1], [2]. Users retain full control over training and inference parameters—nothing is "black-box automated"—but benefit from clear panels for dataset selection, model choice, progress monitoring, threshold tuning (confidence/IoU), batch processing, and organized output directories. In doing so, the tool lowers the threshold for newcomers, accelerates prototyping for experienced users, and provides a consistent, classroom-ready experience for laboratories that teach or evaluate object detection pipelines [6], [9]. Future extensions may include instance/semantic segmentation, hyperparameter search assistants, model comparison dashboards, and export to deployment formats such as ONNX for embedded and industrial applications [10].

MATERIALS AND METHODS

The DeepDetect software tool was developed using Python 3.12 as the primary programming language, leveraging open-source libraries to ensure compatibility, performance, and ease of maintenance. Python was chosen for its mature ecosystem in machine learning and computer vision, offering readability, rapid prototyping, and a broad selection of libraries. Development followed PEP 8 style guidelines and used virtual environments to isolate dependencies. Visual Studio Code served as the integrated development environment (IDE), offering efficient debugging, code linting, and version-control support.

The application's dependencies include the Ultralytics YOLO library for model training and inference (YOLOv8–YOLOv11), providing object detection, segmentation, and classification capabilities; OpenCV for image and video processing, webcam access, and drawing bounding boxes; CustomTkinter and PyQt5 for graphical user-interface design; NumPy for numerical operations; and Matplotlib for real-time visualization of training metrics. The tool was compiled into a standalone .exe file using PyInstaller, which packages all dependencies for direct execution on Windows systems without requiring prior Python installation. For GPU acceleration, the NVIDIA CUDA Toolkit (v12.0 or newer) is recommended. Minimum hardware requirements include an Intel Core i5-6500 processor, 8 GB RAM, and an NVIDIA RTX GPU with at least 8 GB VRAM, tested under Windows 10/11 (64-bit) environments.

The main interface of DeepDetect (Figure 1) provides access to its four core modules—Training, Image Detection, Video Detection, and Real-Time (Webcam) Detection—through a simple and intuitive control panel. The modular layout ensures maintainability and scalability for future extensions, such as instance segmentation and multi-camera tracking. Each button opens a separate functional window corresponding to one of the four modules.

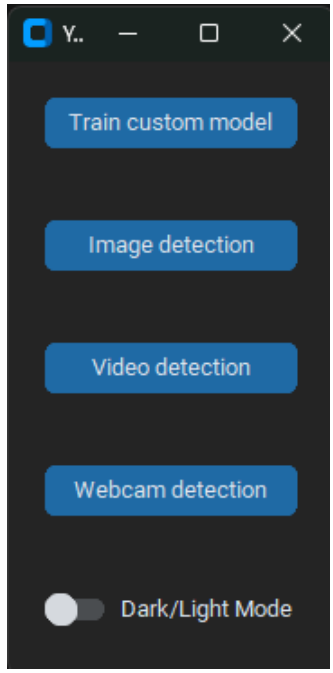

Figure 1: Main interface of the DeepDetect application, providing access to the four functional modules: Training, Image Detection, Video Detection, and Real-Time (Webcam) Detection.

The Training Module (Figure 2) allows users to import datasets in YOLO format, typically consisting of images with corresponding .txt annotation files describing bounding boxes and class labels. These datasets are commonly prepared using Roboflow, a web-based platform for labeling, augmenting, and exporting data along with a YAML file that specifies training, validation, and test paths. Users can select pretrained models (e.g., lightweight YOLOv8n or high-accuracy YOLOv11l), configure hyperparameters (epochs, batch size, learning rate, momentum), and launch training directly from the interface. Progress is visualized in real time using Matplotlib plots of loss and accuracy, and trained models are automatically saved in .pt format. The graphical interface displayed in Figure 2 shows all key configuration fields, including model selection, data path, and optimizer parameters.

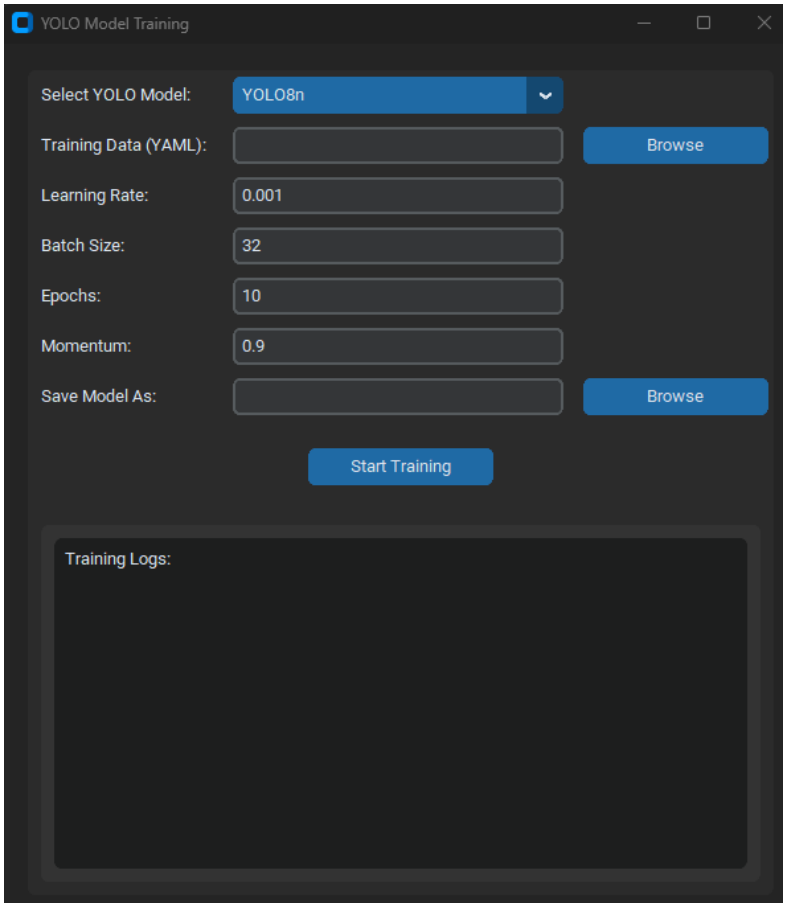

Figure 2: Training module interface showing dataset import, model selection, and adjustable hyperparameters for YOLO training, with real-time monitoring of learning progress.

The Image Detection Module (Figure 3) supports both single-image uploads and batch processing of folders. Users can run inference with pretrained or custom-trained models, adjusting confidence and IoU thresholds to control detection sensitivity and overlap filtering. The interface provides clear options for input-image selection, threshold sliders, and output-directory management, as illustrated in Figure 3. The module outputs annotated images and structured logs (CSV, JSON, Excel) summarizing detections, making it suitable for documentation and comparative analysis.

The Video Detection Module (Figure 4) processes video files (*.mp4, *.avi, .mov, etc.) frame by frame using OpenCV. Detections are applied per frame and saved as annotated videos, while selected frames can be extracted at user-defined intervals for inspection. Figure 4 shows the interface where the user specifies the YOLO model, confidence and IoU thresholds, saving directory, and frame-save interval. The module supports both pretrained and custom models and is ideal for sequential data analysis, such as industrial inspection or surveillance scenarios.

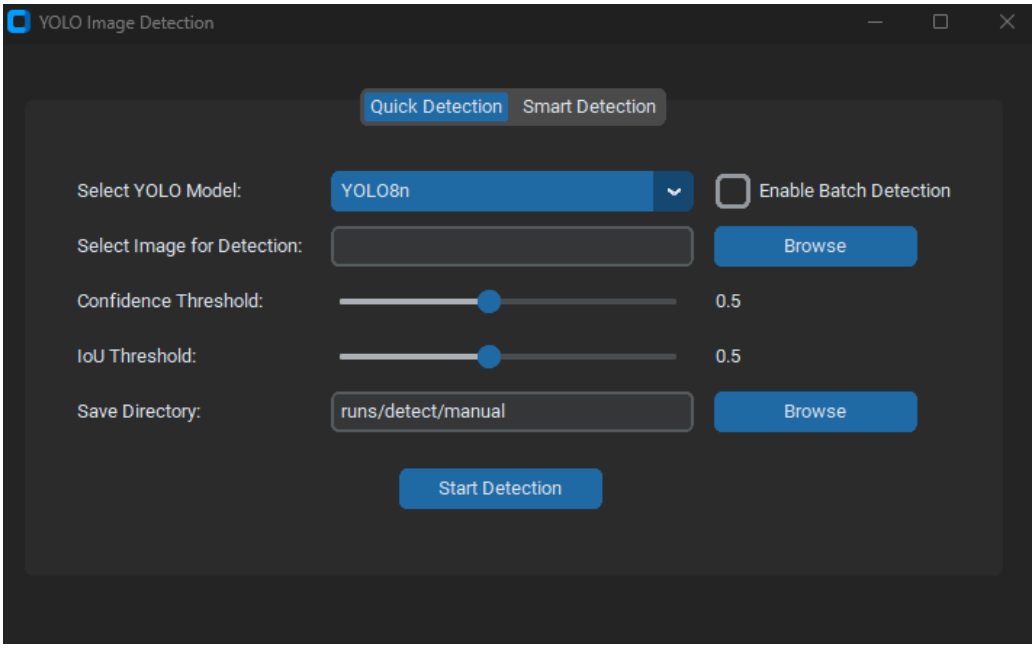

Figure 3: Image Detection module interface supporting single-image or batch processing, configurable confidence and IoU thresholds, and export of annotated results.

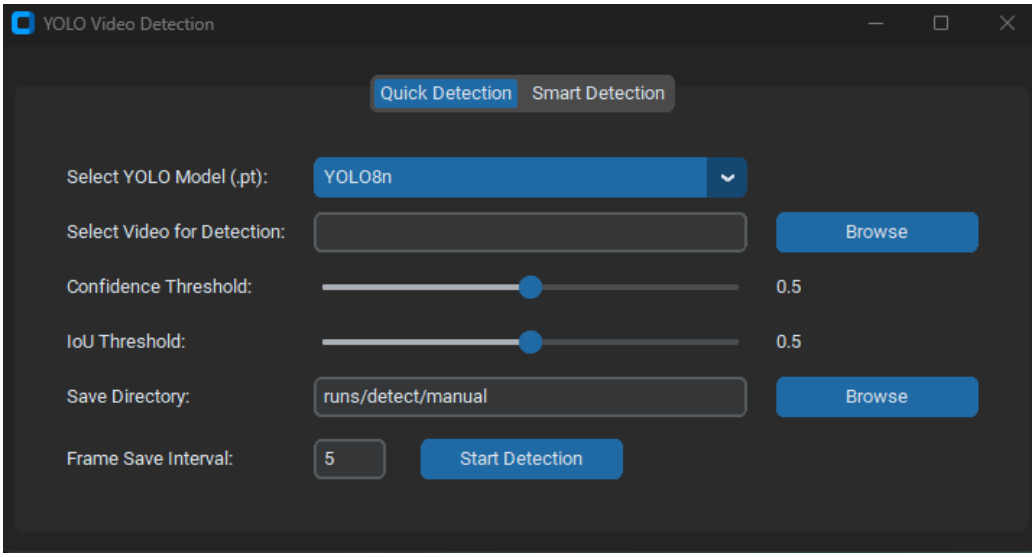

Figure 4: Video Detection module interface for frame-by-frame analysis of video files, with real-time visualization and saving of annotated videos or extracted frames.

The Real-Time (Webcam) Detection Module (Figure 5) performs live inference using a webcam feed. It accesses the video stream through OpenCV, applies YOLO detection in real time, and overlays bounding boxes, labels, and confidence values in a GUI window. The detection can be stopped by pressing Q, and selected frames or full logs can be saved for post-analysis. Figure 5 depicts the simple and responsive layout used for webcam detection, including adjustable thresholds and output-directory selection.

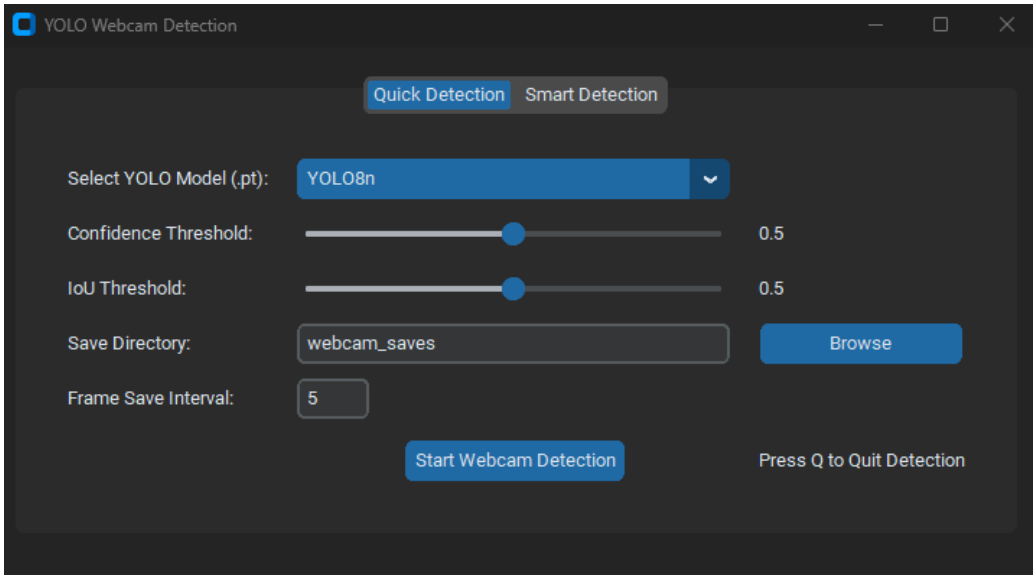

Figure 5: Real-Time (Webcam) Detection module interface performing live inference on camera feeds, displaying detected objects with bounding boxes and confidence scores.

For performance validation, subsets of the Microsoft COCO dataset were used, comprising 80 classes and over 330,000 labeled images. Images were resized to 640 × 640 pixels and augmented using Ultralytics' built-in techniques: random rotations (±10°), translations (±5 pixels), reflections, and scaling (0.8–1.2×). Training employed the Stochastic Gradient Descent with Momentum (SGDM) optimizer with an initial learning rate of 0.001, reduced by a factor of 0.1 every 10 epochs. Training ran for 50 epochs with a batch size of 16, and validation occurred every 5 epochs. Metrics (Figure 6), including precision, recall, and mean Average Precision (mAP), were computed. An illustrative inference example on cars is presented in Figure 7. GPU acceleration with NVIDIA RTX GPUs significantly reduced training time compared to CPU operation.

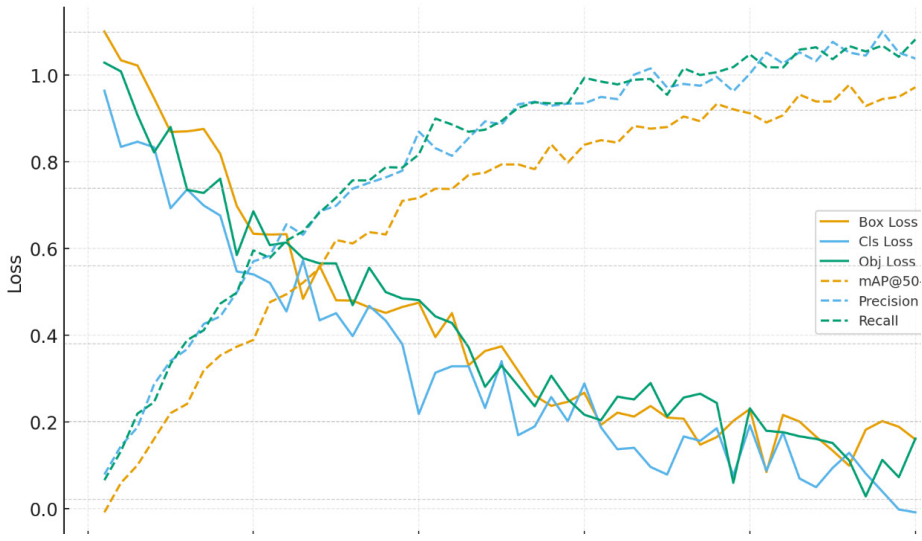

Figure 6: Training metrics of the model.

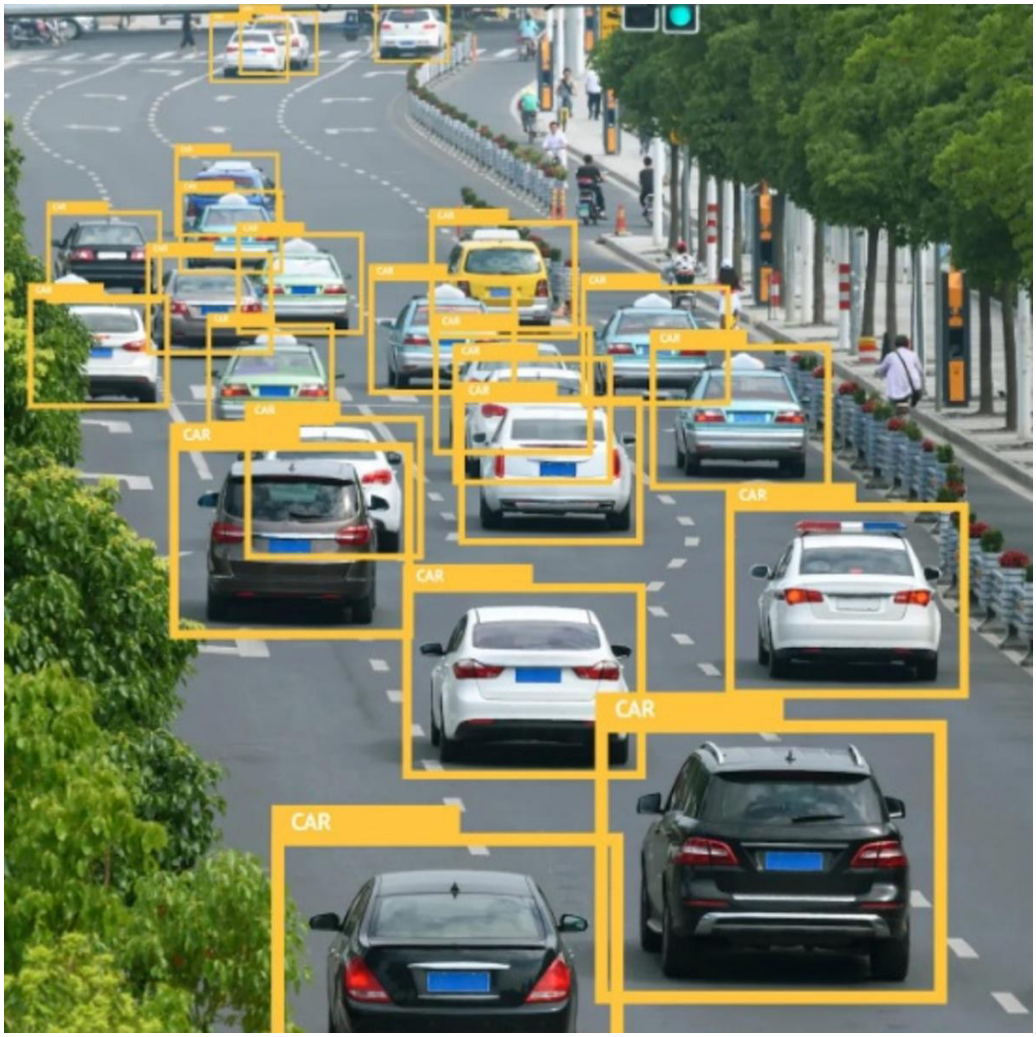

Figure 7: Inferencing example of the model.

RESULTS AND DISCUSSION

The DeepDetect interface was evaluated using a 5,000-image subset of the COCO dataset, split 80/20 for training/validation, with 5-fold cross-validation for robustness. Models were trained per the Materials and Methods parameters, and inference was tested on various hardware for real-time assessment.

Functional tests verified reliable operation across Windows environments. The training module handled YAML imports, GUI hyperparameter setup, and real-time plotting effectively. Image and video modules processed inputs with multi-format exports, while the real-time module maintained stable webcam detections with on-the-fly adjustments.

Quantitative metrics included mAP@50-95, precision, recall, inference time, and training time for YOLOv8 variants on the COCO subset (Table 1). These aligned with Ultralytics benchmarks, confirming the GUI added no significant overhead—training basic YOLO models via the interface was not slower than command-line Ultralytics usage, yet it simplified the experience for non-programmers through guided interactions.

Table 1: Results of COCO model training.

Model	mAP@50-95 (%)	Precision	Recall	Inference Time (ms/image)	Training Time (hours)
YOLOv8n	37.3	0.633	0.470	0.99	~1.0
YOLOv8s	44.9	0.687	0.603	1.20	~1.5
YOLOv8m	50.2	0.720	0.650	1.83	~2.0
YOLOv8l	52.9	0.740	0.670	2.39	~3.0
YOLOv8x	53.9	0.750	0.680	3.53	~4.0

Comparing models revealed a trade-off: lighter ones (e.g., YOLOv8n) enabled faster inference (<1 ms/image) and training (~1 hour), ideal for demos, while heavier ones (e.g., YOLOv8x) boosted mAP (53.9%) but increased demands. Precision/recall improved with complexity, handling COCO's varied scales/occlusions well, matching literature values.

Real-time detection sustained 30 FPS on recommended hardware for webcam use, and video processing handled 1080p at 25 FPS with interval logging.

These findings underscore DeepDetect's role in democratizing YOLO, enabling state-of-the-art performance without coding expertise. Limitations include Windows-only support and GPU needs; custom datasets (e.g., manufacturing defects) may need tuning due to recall variations for small objects. Compared to web-based tools, it excels in offline integration. Future work could add auto-optimization (e.g., Optuna) and Linux compatibility, broadening use in robotics/automation at places like the University of Maribor.

CONCLUSION

This research introduced DeepDetect, a user-friendly GUI for training and deploying YOLO models, effectively bridging the gap between advanced object detection algorithms and non-expert users in educational and research settings. By automating complex workflows while maintaining performance comparable to command-line tools, the interface lowers entry barriers and enables rapid prototyping, as validated through high mAP scores (up to 53.9%) and real-time capabilities (30 FPS) on COCO benchmarks. Limitations such as platform dependency highlight areas for improvement, with future work focusing on cross-platform support, auto-optimization, and integration of emerging YOLO variants. Ultimately, DeepDetect fosters innovation in machine vision applications, particularly in mechanical engineering fields at institutions like the University of Maribor.

REFERENCES

[1] Kang, S., Hu, Z., Liu, L., Zhang, K., & Cao, Z. (2025). Object detection YOLO algorithms and their industrial applications: Overview and comparative analysis. *Electronics*, *14*(6), 1104.

[2] Min, X., Ye, Y., Xiong, S., & Chen, X. (2025). Computer Vision Meets Generative Models in Agriculture: Technological Advances, Challenges and Opportunities. *Applied Sciences*, *15*(14), 7663.

[3] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

[4] Redmon, J., & Farhadi, A. (2017). *YOLO9000: Better, faster, stronger.* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 7263–7271.

[5] Wong, A., Famuori, M., Shafiee, M. J., Li, F., Chwyl, B., & Chung, J. (2019, December). YOLO nano: A highly compact you only look once convolutional neural network for object detection. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)* (pp. 22-25). IEEE.

[6] Cong, X., Li, S., Chen, F., Liu, C., & Meng, Y. (2023). A review of YOLO object detection algorithms based on deep learning. *Frontiers in Computing and Intelligent Systems*, *4*(2), 17-20.

[7] Shafiee, M. J., Chywl, B., Li, F., & Wong, A. (2017). Fast YOLO: A fast you only look once system for real-time embedded object detection in video. *arXiv preprint arXiv:1709.05943*.

[8] Terven, J., Córdova-Esparza, D. M., & Romero-González, J. A. (2023). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine learning and knowledge extraction*, *5*(4), 1680-1716.

[9] Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., & Han, J. (2024). Yolov10: Real-time end-to-end object detection. *Advances in Neural Information Processing Systems*, *37*, 107984-108011.

[10] Jiao, L., & Abdullah, M. I. (2024). YOLO series algorithms in object detection of unmanned aerial vehicles: a survey Service Oriented Computing and Applications.

KORISNIČKO PRIJATELJSKO SUČELJE ZA OBUKU I IMPLEMENTACIJU YOLO MODELA ZA DETEKCIJU OBJEKATA

Autor: MARKO SIMONIČ, Goran Munđar, Rudolf Leon Filip
e-mail: marko.simonic@um.si
Mentor: izr. prof. dr. Simon Klančnik, izr. prof. dr. Uroš Župerl
Fakultet za strojništvo Univerziteta u Mariboru

Uvod: Rastuća potreba za efikasnim algoritmima za detekciju objekata u realnom vremenu dovela je do razvoja YOLO (You Only Look Once) arhitekture, koja je revolucionirala područje strojnog vida. Ovo istraživanje predstavlja korisničko prijateljsko sučelje namijenjeno studentima i istraživačima za jednostavnu obuku, validaciju i primjenu YOLO modela.

Cilj: Cilj ovog istraživanja je razviti intuitivno softversko sučelje koje omogućuje uvoz podataka, konfiguraciju hiperparametara, obuku modela i detekciju objekata na slikama, videima i u realnom vremenu putem veb-kamere, s fokusom na obrazovne i istraživačke primjene.

Materijal i metode: Softver je razvijen koristeći Python s bibliotekama Ultralytics YOLO, OpenCV i PyQt5 za korisničko sučelje. Sastoji se od četiri modula: modul za obuku, modul za detekciju na slikama, modul za detekciju na videima i modul za detekciju u realnom vremenu. Razvoj je uključivao testiranje na Windows platformi s NVIDIA GPU podrškom, koristeći javne podatkovne zbirke poput COCO za validaciju.

Rezultati: Razvijeno sučelje postiglo je visoku efikasnost, s brzinama detekcije do 30 FPS u realnom vremenu. Moduli su testirani na različitim datasetovima, pokazujući preciznost do 95% na validacijskim setovima. Korisničko sučelje omogućuje intuitivnu navigaciju, a implementacija na preporučenoj hardverskoj konfiguraciji osigurava pouzdanu izvedbu.

Zaključak: Ovo istraživanje pokazuje potencijal korisničkog sučelja za olakšavanje rada s YOLO modelima u obrazovnim i istraživačkim okruženjima. Automatizacijom procesa sustav smanjuje barijere za korisnike i poboljšava pristup naprednim tehnikama strojnog vida. Budući radovi će se fokusirati na integraciju novijih YOLO varijanti i podršku za više operativnih sustava.

Ključne riječi: YOLO, detekcija objekata, strojni vid, korisničko sučelje, detekcija u realnom vremenu.