Real-Time Weapon Detection Using CNN-Based Model

Pratik Naik

Department of Computer Science, Pace University – New York City Campus

CS672: Introduction to Deep Learning

Professor Omar Mostafa

May 10, 2025

Abstract: Real-Time Weapon Detection Using CNN-Based Model

The increasing prevalence of violence in public spaces has accelerated the demand for intelligent surveillance systems capable of real-time threat detection. This project presents the design and implementation of a real-time weapon detection system using a custom Convolutional Neural Network (CNN) model, developed without relying on pre-trained architectures. By leveraging fundamental deep learning principles, the system is trained to identify weapons—specifically guns, pistols, knives, and grenades—within a YOLOv5-annotated image dataset. The model's architecture comprises stacked convolutional and pooling layers optimized for spatial feature extraction, followed by fully connected layers for multi-class classification. The trained model is integrated with OpenCV to enable real-time object detection through a laptop webcam, drawing bounding boxes and generating terminal-based alerts upon detecting a weapon. While the model achieves moderate accuracy, the project underscores the feasibility of deploying lightweight CNNs for live security applications with constrained hardware. This report outlines the dataset preprocessing methods, model architecture, training pipeline, and live implementation, concluding with a discussion on limitations and avenues for future enhancements such as dataset expansion, model optimization, and hardware-based alert integration.

Keywords: Weapon Detection, Convolutional Neural Networks (CNN), Real-Time Surveillance, Object Detection, Deep Learning, OpenCV, YOLOv5 Format, Image Classification, Security Applications, Custom Model Architecture, Live Video Inference, Threat Identification

Literature Review

The field of computer vision has witnessed rapid growth with the advent of deep learning techniques, particularly Convolutional Neural Networks (CNNs), which have become the foundation for numerous object detection and image classification tasks. Early approaches in image recognition relied heavily on handcrafted features and traditional classifiers such as Support Vector Machines (SVMs) or k-Nearest Neighbors (k-NN), which often lacked robustness in complex environments (LeCun et al., 2015). The introduction of CNNs revolutionized visual recognition by enabling automatic feature extraction through hierarchical learning, leading to breakthroughs in tasks like face recognition, traffic sign detection, and object localization (Krizhevsky, Sutskever, & Hinton, 2012).

Object detection models such as R-CNN, Fast R-CNN, and YOLO (You Only Look Once) have established strong benchmarks in real-time visual analysis by combining localization and classification in unified frameworks (Redmon et al., 2016). While these models achieve high accuracy, they often rely on pre-trained architectures and large computational resources, which may not be ideal for lightweight or real-time deployment scenarios. Hence, there is increasing interest in building custom models that are tailored for specific tasks and optimized for resource-constrained environments.

Real-time weapon detection, a subdomain of surveillance-based object detection, is of significant societal importance due to rising concerns around public safety. Research in this area has explored both classical and deep learning methods. Traditional approaches involving motion detection or shape matching have shown limited success in cluttered or low-light environments. Modern solutions leverage datasets formatted using standards like YOLOv5, which provide structured annotations for bounding box detection and class labels (Bochkovskiy et al., 2020).

CNNs, when trained from scratch using labeled image datasets, can yield competitive performance for domain-specific detection tasks such as weapon recognition. However, challenges remain in terms of dataset diversity, model generalization, and false positives in uncontrolled environments. Techniques such as dropout regularization, data augmentation, and model pruning have been employed to mitigate overfitting and enhance model robustness (Srivastava et al., 2014).

Additionally, real-time deployment necessitates efficient integration with tools such as OpenCV for webcam interfacing and TensorFlow or Keras for inference. Studies highlight the need for pipeline optimization, including frame resizing, class activation mapping, and bounding box drawing, to achieve a balance between detection speed and accuracy (Rosebrock, 2018). As deep learning becomes more accessible, academic projects have increasingly focused on implementing such systems using minimal resources, demonstrating the feasibility of practical AI-powered surveillance without reliance on proprietary or heavy-weight architectures.

This literature review contextualizes the current project, which seeks to build a lightweight, real-time weapon detection system using a CNN trained from scratch, emphasizing the academic relevance and real-world applicability of such deep learning-based computer vision solutions.

Programming Environment

The development of deep learning applications such as real-time object detection relies heavily on a cohesive software stack that supports rapid experimentation, high computational efficiency, and ease of deployment. In this project, a custom Convolutional Neural Network (CNN) was implemented and trained using TensorFlow and Keras—two of the most widely adopted deep learning frameworks in academic and industrial settings. The detection pipeline was integrated with OpenCV, enabling real-time video processing through a laptop webcam. The overall programming environment was chosen to balance accessibility, modularity, and compatibility with system hardware, particularly for local execution on a MacBook Pro with M1 architecture.

TensorFlow and Keras

TensorFlow, developed by Google, is a powerful open-source platform for numerical computation and large-scale machine learning. Keras, which operates as an API within TensorFlow, allows rapid prototyping and model design with a high-level interface. Together, they provide an ideal development environment for building and training custom CNNs, supporting essential functionalities such as layer configuration, loss function selection, optimizer integration, and model evaluation.

Key Features:

- High-level API for building sequential or functional neural networks.
- GPU and M1-compatible backend execution via TensorFlow-Metal and TensorFlow-Mac.
- Inbuilt utilities for data preprocessing, augmentation, and performance tracking.
- Model serialization formats such as .h5 for deployment and inference.

Advantages:

- Extensive community support and documentation.
- Seamless integration with NumPy and Pandas for dataset handling.
- Real-time debugging using TensorBoard.

Limitations:

- Memory management and low-level control are abstracted, which may limit fine-tuned optimization.
- M1 support, while improving, still has limited compatibility with some TensorFlow operations.

OpenCV (Open Source Computer Vision Library)

OpenCV plays a pivotal role in enabling real-time video capture, frame preprocessing, and overlay visualization. In this project, it serves as the interface between the webcam feed and the trained CNN model, facilitating frame-wise prediction and bounding box rendering.

Key Features:

- Real-time video capture and frame processing from external or in-built cameras.
- Image transformation utilities such as resizing, color space conversion, and drawing functions.
- Easy integration with NumPy arrays for direct input into TensorFlow models.

Advantages:

- Cross-platform compatibility and lightweight deployment.
- Extensive support for video I/O, making it ideal for live applications.

Limitations:

- Manual tuning required for performance in low-light or cluttered environments.
- Lack of GPU acceleration unless explicitly configured with third-party bindings.

Supporting Libraries

Additional libraries including Matplotlib and Seaborn were employed for data visualization (e.g., training accuracy and loss curves), while Pandas and Scikit-learn facilitated dataset handling and model evaluation. These tools collectively streamline the development lifecycle, from preprocessing and training to live testing and evaluation.

Applications

The development and deployment of real-time weapon detection systems represent a critical intersection of deep learning, computer vision, and public safety. Convolutional Neural Networks (CNNs), when trained on relevant image datasets, can classify and localize objects within visual data, making them suitable for intelligent surveillance and threat monitoring systems. This project demonstrates the viability of such systems by applying a custom CNN to live webcam feeds for weapon identification. Below are key domains where real-time object detection and similar CNN-based solutions have transformative applications:

- **Security and Surveillance**: One of the most direct applications of real-time object detection is in public safety and surveillance. Weapon detection in environments such as schools, airports, and government buildings can enable automated alerts, reducing response time during potential threats. Integrating deep learning models into CCTV or mobile security systems enhances situational awareness and decision-making in real time.
- **Law Enforcement and Public Safety**: Police departments and security agencies can benefit from AI-powered systems that identify concealed or exposed weapons during patrols, border inspections, or large public events. These models can assist in monitoring live body cam footage or dash cam video streams, providing critical assistance to officers in high-risk scenarios.
- **Military and Defense**: Autonomous surveillance drones and unmanned ground vehicles (UGVs) can use CNN-based object detection models to scan for weapons or suspicious activities in restricted areas. This reduces reliance on manual monitoring and increases operational intelligence in combat or surveillance missions.

- **Smart Infrastructure and Transportation Hubs**: Real-time detection systems can be embedded in smart infrastructure at transportation hubs such as train stations, subways, and airports to preemptively flag threats. When deployed alongside facial recognition or behavioral analysis systems, object detection adds an essential layer of risk assessment.
- **Retail and Commercial Premises**: Increasingly, retail outlets, banks, and service centers are investing in AI-driven security systems to detect shoplifting or weapon-related threats. Custom models like the one developed in this project can be adapted to commercial environments where pre-trained models may not capture context-specific objects.
- **Academic and Research Environments**: Projects such as this also serve as academic case studies in the implementation and optimization of custom CNN architectures. They are vital for students and researchers exploring lightweight deep learning pipelines suitable for real-time applications under hardware constraints.

This application landscape demonstrates that lightweight CNNs, even when built without pre-trained backbones, can provide actionable intelligence across multiple domains where safety and surveillance are priorities.

Challenges

While deep learning-based object detection provides a promising pathway for enhancing real-time surveillance systems, it also presents several technical and deployment challenges—especially when using custom-built CNN architectures rather than pre-trained models. These challenges influence both the training and real-time inference phases and require careful consideration to ensure practical applicability and performance.

- **Model Accuracy and Generalization**: One of the foremost challenges in building a custom CNN for weapon detection is ensuring that the model generalizes well across diverse lighting conditions, occlusions, angles, and backgrounds. Limited datasets can lead to overfitting, where the model performs well on training data but poorly in real-world environments.
- **Real-Time Inference Performance**: Achieving smooth, low-latency detection using webcam input can be demanding, especially on hardware with limited GPU acceleration. Frame capture, preprocessing, model inference, and visualization must occur within milliseconds to avoid lag, requiring efficient code optimization and lightweight model design.
- **Dataset Limitations and Annotation Quality**: The quality of the dataset significantly impacts model performance. In weapon detection, mislabeling or inadequate diversity in training samples can lead to high false positive or false negative rates. Furthermore, preparing a YOLOv5-style annotated dataset manually is time-consuming and prone to inconsistencies.

- **Hardware Constraints**: Running inference in real time without relying on cloud GPUs or high-end hardware limits the complexity of the model. Devices such as consumer laptops, especially those with integrated or M1 GPUs, may not support the full capabilities of TensorFlow, requiring compatibility workarounds and reduced batch sizes.
- **Integration and Visualization Complexity**: Combining the trained model with OpenCV for webcam-based detection introduces its own set of integration challenges. Proper synchronization between frame capture and model prediction, along with clear visualization of bounding boxes and alerts, demands careful pipeline orchestration.
- **Security and Reliability in Deployment**: For systems designed to assist in public safety, ensuring robust performance is critical. The risk of missed detections or erroneous classifications can compromise the system's utility. Additionally, fail-safe mechanisms and alert reliability must be considered when integrating into larger security infrastructures.

Despite these challenges, the implementation of a custom CNN-based weapon detection system remains a viable proof of concept. It highlights key considerations for future enhancements, including expanded datasets, model pruning, hardware-specific optimization, and potential integration with cloud-based services for scalability.

Conclusion

The implementation of a real-time weapon detection system using a custom Convolutional Neural Network (CNN) demonstrates the practical potential of deep learning in safety-critical applications. This report outlined the design and development of a lightweight CNN trained on a YOLOv5-annotated dataset, capable of identifying weapons such as guns, pistols, knives, and grenades through a live webcam feed. Key components of the project included data preprocessing, architecture construction, training visualization, and integration with OpenCV for real-time inference and alert generation.

Despite its simplicity, the model successfully establishes a foundation for deploying AI-powered surveillance tools on consumer-grade hardware. The report also addressed technical challenges, including dataset limitations, inference latency, and integration hurdles, providing insight into the trade-offs involved when developing models from scratch versus using pre-trained alternatives.

This exploration highlights the growing accessibility and relevance of custom deep learning models in enhancing public safety systems. As the field evolves, future improvements may include expanding dataset diversity, leveraging model optimization techniques such as quantization and pruning, and deploying models on edge or cloud platforms for greater scalability and performance. Ultimately, this project reinforces the importance of applied deep learning in real-world security domains and paves the way for more intelligent, responsive, and automated surveillance solutions.

References

Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). *YOLOv4: Optimal speed and accuracy of object detection*. arXiv. https://arxiv.org/abs/2004.10934

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Communications of the ACM, 60*(6), 84–90. https://doi.org/10.1145/3065386

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Rosebrock, A. (2018). *Deep learning for computer vision with Python* (Vol. 1). PyImageSearch.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*(1), 1929–1958.

TensorFlow. (n.d.). *TensorFlow documentation*. https://www.tensorflow.org/