

REAL-TIME PEDESTRIAN DETECTION USING YOLOV8 WITH OPENVINO OPTIMIZATION

Md Maheub Ali^{1*}, Patuwardhanam Sai Srikar², Pundru Saideep², Shruti Balivada²

¹Assistant Professor, ²UG Student, ^{1,2}Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning)

^{1,2}J.B. Institute of Engineering and Technology

*Corresponding author: Md Maheub Ali (maheub.ali24@gmail.com)

ABSTRACT

Pedestrian detection plays a vital role in modern intelligent transportation systems, autonomous driving, and video surveillance applications, where accurate and real-time object detection is essential for ensuring safety and efficiency. In this paper, we propose a robust and efficient pedestrian detection system based on the YOLOv8 (You Only Look Once version 8) deep learning model, further optimized using the OpenVINO toolkit to enhance inference speed and deployment performance. The proposed approach leverages the high accuracy of YOLOv8 for object detection and combines it with hardware-aware optimizations to achieve real-time processing capabilities. The model is initially trained on a pedestrian focused dataset to accurately identify human instances in diverse environmental conditions, including varying lighting, occlusions, and complex backgrounds. After training, the model is converted into OpenVINO's Intermediate Representation (IR) format and optimized using techniques such as INT8 quantization and layer fusion. These optimizations significantly reduce computational complexity and improve throughput without causing substantial degradation in detection accuracy. To make the system accessible and user-friendly, a web-based application is developed using the Flask framework, enabling users to upload video streams or images and visualize detection results in real time. The backend integrates the optimized model to process inputs efficiently, while the frontend provides an interactive interface for seem less user experience.

Experimental evaluation demonstrates that the proposed system achieves a notable increase in frames per second (FPS) and reduced latency compared to the baseline YOLOv8 model, making it highly suitable for deployment in edge devices and real-time monitoring systems. The results highlight the effectiveness of combining state-of-the-art deep learning models with inference optimization frameworks to bridge the gap between accuracy and performance. Overall, the proposed system provides a scalable and efficient solution for real-world pedestrian detection applications.

Key Words: PedestrianDetection, YOLOv8, OpenVINO, Deep Learning, Computer Vision, Real-Time Detection, Object Detection, INT8 Quantization, Edge Computing, Flask.

1.INTRODUCTION

In recent years, pedestrian detection has emerged as a fundamental task in the field of computer vision, with significant applications in intelligent transportation systems, autonomous driving, smart surveillance, and public safety monitoring. As urban environments become increasingly complex and densely populated, the need for accurate and real-time detection of pedestrians has grown substantially. Effective pedestrian detection systems can help reduce road accidents, enhance driver assistance systems, and improve situational awareness in automated systems. However, achieving reliable detection in real-world scenarios remains a challenging problem due to variations in lighting conditions, scale, pose, occlusions, and cluttered backgrounds.

Earlier approaches to pedestrian detection primarily relied on traditional computer vision techniques that utilized handcrafted feature extraction methods such as Histogram of Oriented Gradients (HOG), Haar-like features, and edge-based descriptors, combined with machine learning classifiers like Support Vector Machines (SVM) and AdaBoost. Although these methods provided reasonable performance in controlled environments, they often failed to generalize effectively in complex and dynamic scenes, limiting their applicability in real-time systems. The advancement of deep learning, particularly convolutional neural networks (CNNs), has revolutionized object detection by enabling automatic feature learning and significantly improving detection accuracy. Among various deep learning-based object detection frameworks, the YOLO (You Only Look Once) family has gained widespread attention due to its unified architecture and real-time detection capability. Unlike two stage detectors such as R-CNN, YOLO-based models perform detection in a single forward pass, making them highly efficient for time sensitive applications. In this work, we adopt YOLOv8, a recent and improved version of the YOLO architecture, which offers enhanced accuracy, better feature representation, and optimized performance compared to its predecessors. Despite the high accuracy of YOLOv8, deploying deep learning models in real-time applications, especially on resource-constrained devices such as edge systems, remains a significant challenge. High computational requirements, increased memory usage, and latency issues can hinder practical implementation. To overcome these limitations, this paper incorporates the OpenVINO (Open Visual Inference and Neural Network Optimization) toolkit, which is designed to optimize deep learning models for efficient inference on Intel hardware. The trained YOLOv8 model is converted into OpenVINO's Intermediate Representation (IR) format and

further optimized using techniques such as model compression, graph optimization, and INT8 quantization. These optimizations reduce model size and computational complexity while maintaining a balance between speed and detection accuracy.

In addition to model optimization, the usability of the system is enhanced through the development of a web-based interface using the Flask framework. The application enables users to interact with the system by uploading images or video streams and obtaining real-time pedestrian detection outputs. This integration bridges the gap between complex deep learning models and practical deployment, making the system accessible for real-world applications such as traffic monitoring, smart surveillance, and security systems. The proposed system is evaluated based on key performance metrics, including detection accuracy, inference speed measured in frames per second (FPS), and latency. Experimental results demonstrate that the optimized model achieves a significant improvement in inference speed compared to the baseline YOLOv8 model, while maintaining competitive detection performance. This highlights the effectiveness of combining advanced deep learning architectures with hardware-aware optimization techniques for real-time applications.

The main contributions of this paper can be summarized as follows:

1. Design and implementation of an accurate pedestrian detection system using YOLOv8.
2. Integration of OpenVINO for model optimization and accelerated inference.
3. Application of INT8 quantization to improve computational efficiency with minimal accuracy loss.
4. Development of a Flask-based web interface for real-time interaction and visualization.

2.LITERATURE SURVEY

Pedestrian detection has been extensively studied in the field of computer vision due to its importance in applications such as autonomous driving, surveillance systems, and intelligent transportation. Over the years, various approaches have been proposed, ranging from traditional machine learning techniques to advanced deep learning-based models. Early methods for pedestrian detection relied on handcrafted feature extraction techniques combined with classical classifiers. One of the most influential approaches was the use of Histogram of Oriented Gradients (HOG) features along with Support Vector Machines (SVM), which demonstrated good performance in detecting humans in images. Similarly, Haar-like features with AdaBoost classifiers were also widely used for object detection tasks. Although these approaches were computationally efficient, they struggled to handle complex scenarios involving occlusions, varying poses, and dynamic backgrounds, limiting their effectiveness in real-world applications.

With the advancement of deep learning, convolutional neural networks (CNNs) significantly improved object detection accuracy by automatically learning hierarchical feature representations. Region based Convolutional Neural Networks (RCNN) and its variants, such as Fast R-CNN and Faster R-CNN, introduced a two-stage detection pipeline that first generates region proposals and then classifies them. These methods achieved high accuracy but were computationally expensive and not suitable for real-time applications. To address the limitations of two-stage detectors, single-stage detectors such as SSD (Single Shot Multi-Box Detector) and YOLO (You Only Look Once) were introduced. The YOLO framework revolutionized object detection by performing detection in a single forward pass, making it

significantly faster while maintaining competitive accuracy. Subsequent versions, including YOLOv3, YOLOv4, and YOLOv5, introduced improvements in feature extraction, multiscale detection, and training strategies. The latest version, YOLOv8, further enhances performance by incorporating architectural optimizations and improved loss functions, making it highly suitable for real-time detection tasks. In addition to model architecture advancements, recent research has focused on optimizing deep learning models for deployment in resource constrained environments. Techniques such as model pruning, quantization, and hardware-specific optimizations have been widely explored. The OpenVINO toolkit has gained attention as an effective framework for optimizing and deploying deep learning models on Intel hardware. It enables model conversion into an intermediate representation and supports precision reduction techniques such as INT8 quantization, which significantly improves inference speed and reduces memory consumption.

Several studies have demonstrated the effectiveness of combining YOLO-based models with optimization frameworks like OpenVINO for real-time applications. These approaches achieve a balance between detection accuracy and computational efficiency, making them suitable for edge devices and embedded systems. However, many existing works focus primarily on model performance and lack user-friendly interfaces for practical deployment. In contrast, the proposed system not only utilizes YOLOv8 for accurate pedestrian detection but also integrates OpenVINO for optimized inference and incorporates a Flask-based web application for real-time user interaction. This combination addresses both performance and usability aspects, making the system more practical for real world deployment scenarios such as surveillance and traffic monitoring.

3.PROPOSED SYSTEM

The proposed system aims to develop an efficient and real-time pedestrian detection framework by integrating a state-of-the-art deep learning model with hardware-aware optimization techniques and a user-friendly web interface. The overall architecture consists of four major components: data preprocessing and model training, model optimization using OpenVINO, backend integration, and frontend visualization.

The overall architecture consists of four major components: data preprocessing and model training, model optimization using OpenVINO, backend integration, and frontend visualization. The system is designed to ensure high detection accuracy while maintaining low latency and improved inference speed suitable for real world deployment.

Pedestrian Detection using YOLOv8 with OpenVINO Optimization

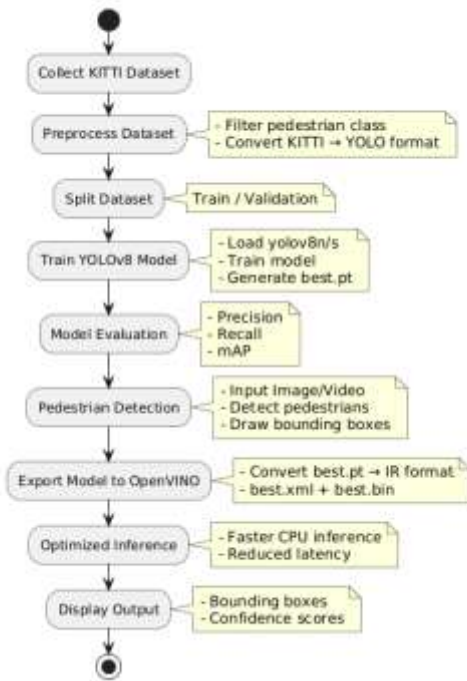


Figure 3.1 Proposed Methodology

3.1 System Overview

The workflow of the proposed system begins

with input data in the form of images or video streams. These inputs are processed by the trained YOLOv8 model to detect pedestrians by generating bounding boxes, confidence scores, and class labels. The trained model is then optimized using the OpenVINO toolkit to enhance inference performance. The optimized model is deployed within a Flask-based web application, allowing users to interact with the system and visualize detection results in real time.

3.2 Data Collection and Preprocessing

A pedestrian-focused dataset is used to train the detection model. The dataset consists of annotated images containing pedestrian instances under varying environmental conditions, such as different lighting, occlusions, and crowd densities. Preprocessing steps include image resizing, normalization, and data augmentation techniques such as flipping, scaling, and rotation to improve model generalization and robustness. The dataset is divided into training and validation sets to ensure proper evaluation of model performance. This structured dataset preparation enables the model to learn diverse patterns and improves its ability to detect pedestrians accurately in real-world scenarios. Additionally, proper annotation quality ensures precise localization of objects during training.

3.3 Model Training using YOLOv8

The YOLOv8 model is employed for pedestrian detection due to its efficiency and accuracy in real-time object detection tasks. The model is trained using labeled data to learn spatial and semantic features of pedestrians. During training, the model optimizes loss functions related to bounding box regression, objectness score, and classification.

The output of the model includes bounding boxes around detected pedestrians along with confidence scores. The trained weights (best.pt)

represent the learned parameters and are used for inference.

3.4 Model Optimization using OpenVINO

To address the computational challenges associated with deploying deep learning models, the trained YOLOv8 model is optimized using the OpenVINO toolkit. The optimization process involves converting the trained model into OpenVINO's Intermediate Representation (IR) format, which consists of XML and BIN files. Further optimizations include graph simplification, layer fusion, and precision reduction. One of the key techniques used is INT8 quantization, which reduces the precision of model parameters from floating-point (FP32) to 8-bit integers. This significantly decreases memory usage and improves inference speed while maintaining acceptable accuracy. These optimizations enable the system to run efficiently on edge devices and CPUs without requiring high end GPUs.

3.5 Backend Implementation

The backend of the system is developed using the Flask framework, which acts as a bridge between the user interface and the optimized detection model. The backend handles user inputs, processes images or video streams, performs inference using the optimized model, and returns the detection results. It also manages API frontend and the model, ensuring smooth data flow and real-time processing.

3.6 Frontend Interface

A web-based frontend interface is designed to provide an interactive and user-friendly experience. Users can upload images or videos through the interface and view the detection results with bounding boxes highlighting pedestrians. The interface is built using standard web technologies such as HTML, CSS, and JavaScript, ensuring accessibility and ease of use.

3.7 Performance Evaluation

The performance of the proposed system is evaluated based on key metrics such as accuracy, precision, recall, and inference speed measured in frames per second (FPS). The optimized model demonstrates a significant improvement in FPS compared to the baseline YOLOv8 model, making it suitable for real-time applications. Additionally, the system maintains a good balance between speed and accuracy, which is critical for deployment in practical scenarios.

4.RESULT DESCRIPTION

The performance of the proposed pedestrian detection system is evaluated based on key metrics such as detection accuracy, precision, recall, and inference speed measured in frames per second (FPS). The evaluation is conducted on a diverse set of test images and video sequences containing pedestrians under varying environmental conditions, including different lighting scenarios, partial occlusions, and crowded backgrounds. The results demonstrate the effectiveness of the proposed approach in achieving a balance between accuracy and real-time performance.

4.1 Detection Performance

The YOLOv8 model exhibits strong detection capabilities, accurately identifying pedestrians in both simple and complex scenes. The model successfully detects multiple pedestrians within a single frame, even in cases where individuals are partially occluded or located at different scales. The bounding boxes generated by the model are well aligned with the pedestrian regions, and the confidence scores indicate reliable predictions. The use of data augmentation during training further improves the model's robustness and generalization ability across diverse scenarios. Additionally, the model maintains consistent performance across varying environmental conditions, including changes in lighting and

background complexity. This consistency highlights the effectiveness of the model in real-world applications where conditions are dynamic and unpredictable.

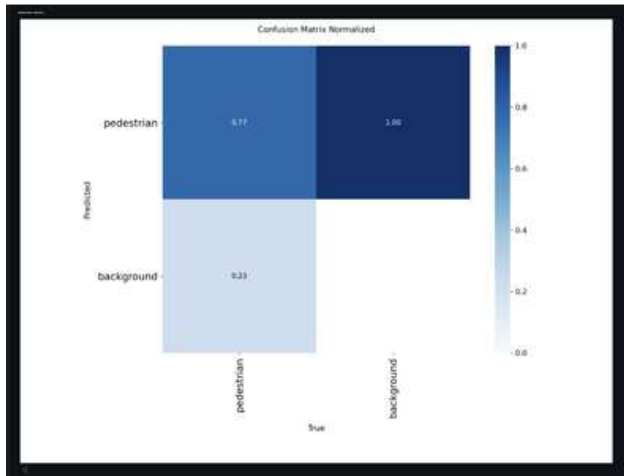


Figure 4.1.1 Normalized Confusion Matrix for Pedestrian Detection Model Showing Classification Performance between Pedestrian and Background Classes

Figure 4.4.1 shows the normalized confusion matrix of the proposed pedestrian detection model, representing its classification performance between pedestrian and background classes. The matrix indicates that the model correctly identifies a significant proportion of pedestrian instances, as reflected by the high true positive rate. However, a small percentage of misclassifications can be observed, where some pedestrian instances are incorrectly classified as background. Similarly, the model demonstrates strong performance in distinguishing background regions, although minor false positives are present. Overall, the confusion matrix highlights the effectiveness of the model in accurately detecting pedestrians while maintaining a reasonable balance between precision and recall.

4.2 Impact of OpenVINO Optimization

A significant improvement in inference performance is observed after applying OpenVINO optimization techniques. The conversion of the trained model into the

Intermediate Representation (IR) format, along with graph optimization and layer fusion, reduces computational overhead. Furthermore, INT8 quantization decreases model precision from FP32 to 8-bit integers, resulting in faster computations and lower memory usage. Comparative analysis between the baseline YOLOv8 model and the optimized model shows a notable increase in FPS, demonstrating the effectiveness of hardware-aware optimizations. While a slight reduction in accuracy may occur due to quantization, the impact is minimal and acceptable for real-time applications. The optimized model is capable of running efficiently on CPU-based systems without requiring high-end GPU resources.

4.3 Inference Speed and Latency

Inference speed is a critical factor for real-time pedestrian detection systems. The baseline YOLOv8 model provides moderate FPS, but its performance may not be sufficient for deployment in latency-sensitive environments. After optimization with OpenVINO, the system achieves a significant increase in FPS, enabling smooth real-time processing of video streams. Additionally, latency is reduced, ensuring faster response times for detection outputs. The improved performance makes the system suitable for applications such as traffic monitoring, surveillance systems, and edge-based deployments, where computational resources are limited and real-time processing is essential.

4.4 Web Application Performance

The Flask-based web application provides an interactive interface for users to upload images and video streams for pedestrian detection. The integration of the optimized model ensures that detection results are displayed with minimal delay. The system effectively handles user requests and processes inputs in real time, demonstrating the practicality of deploying deep learning models through web-based platforms. The frontend interface clearly visualizes

detection outputs by displaying bounding boxes around pedestrians, enhancing user understanding and usability. The seamless communication between frontend and backend through API calls ensures efficient data flow and system responsiveness.

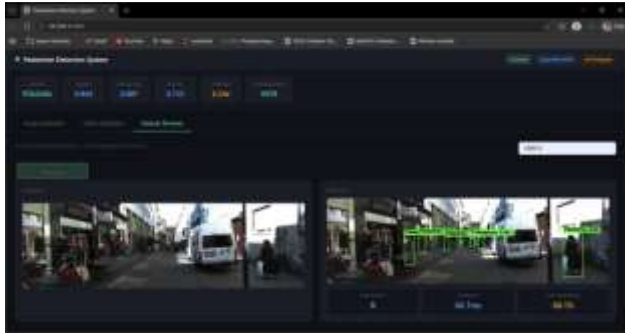


Figure 4.4.1 Dataset Browser Interface with Pedestrian Detection Results using YOLOv8 and OpenVINO Optimization

Figure 4.4.1 shows the dataset browser interface of the proposed pedestrian detection system, where images from the KITTI validation dataset are selected and processed. The figure presents both the original input image and the corresponding detection output, in which multiple pedestrians are accurately identified using bounding boxes. The system demonstrates its ability to detect pedestrians in complex urban environments with varying scales, occlusions, and background conditions. Additionally, performance metrics such as the number of detected pedestrians, inference time, and average confidence score are displayed, highlighting the effectiveness of the optimized model.

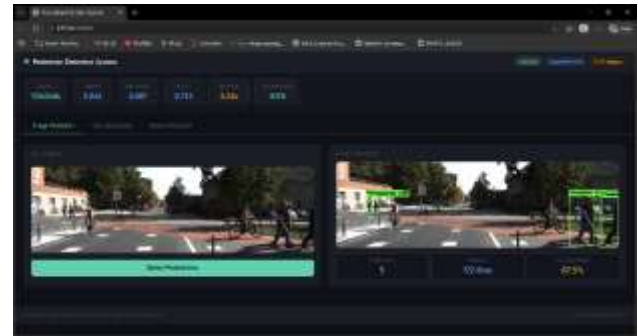


Figure 4.4.2 Image-Based Pedestrian Detection Output Showing Bounding Boxes, Inference Time, and Confidence Scores

Figure 4.4.2 shows the image-based pedestrian detection module of the web application, where a user-provided input image is processed in real time. The detection results clearly illustrate bounding boxes around pedestrians along with confidence scores, enabling easy visualization and interpretation. The system efficiently identifies multiple pedestrians in the scene while maintaining low inference latency. This demonstrates the practical usability of the proposed system in real world applications, where users can interact with the model through a simple and intuitive interface.



Figure 4.4.3 Comparison of YOLOv8 Performance Using PyTorch FP32, OpenVINO FP32, and OpenVINO INT8 in Terms of Inference Time (ms) and Frames Per Second (FPS) on CPU

Figure 4.4.3 shows the performance comparison of the pedestrian detection model across different implementations, namely PyTorch FP32, OpenVINO FP32, and OpenVINO INT8. The results indicate that the OpenVINO INT8 optimized model significantly reduces inference

time while achieving a substantial increase in frames per second (FPS). Compared to the baseline PyTorch implementation, the optimized model demonstrates improved computational efficiency, making it suitable for real-time deployment on CPU-based systems. This highlights the effectiveness of model optimization techniques in enhancing performance without significantly compromising detection accuracy.

4.6 Summary of Results

Overall, the experimental results demonstrate that the proposed system: Achieves high detection

1. Achieves high detection accuracy using YOLOv8.
2. Significantly improves inference speed through OpenVINO optimization.
3. Maintains a balance between accuracy and computational efficiency.
4. Provides real-time performance suitable for practical deployment. Offers a user-friendly interface for interaction and visualization.

5.CONCLUSION

In this paper, an efficient pedestrian detection system based on YOLOv8 and OpenVINO optimization is presented. The proposed approach successfully combines the high accuracy of deep learning with hardware aware optimization techniques to achieve real-time performance. The application of INT8 quantization significantly reduces inference time and improves frames per second (FPS), making the system suitable for deployment on CPU-based and edge devices. Additionally, the integration of a Flask-based web interface enhances usability by enabling real-time interaction and visualization. Experimental results demonstrate that the proposed system achieves a good balance between accuracy and

computational efficiency. Future work can focus on further improving detection accuracy in highly crowded scenarios and extending the system for multi-class object detection.

6.REFERENCES

- [1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. CVPR, 2016.
- [2] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in Proc. CVPR, 2017.
- [3] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv preprint arXiv:1804.02767, 2018.
- [4] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv preprint arXiv:2004.10934, 2020.
- [5] G. Jocher et al., "YOLOv5," GitHub Repository, 2020.
- [6] Ultralytics, "YOLOv8 Documentation," 2023. [Online]. Available: <https://docs.ultralytics.com>
- [7] R. Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in Proc. CVPR, 2014.
- [8] R. Girshick, "Fast R-CNN," in Proc. ICCV, 2015.
- [9] S. Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in Proc. NIPS, 2015.
- [10] W. Liu et al., "SSD: Single Shot MultiBox Detector," in Proc. ECCV, 2016.
- [11] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in Proc. CVPR, 2005.

- [12] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in Proc. CVPR, 2001.
- [13] Intel Corporation, "OpenVINO Toolkit Documentation," 2023. [Online]. Available: <https://docs.openvino.ai>
- [14] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in Proc. NIPS, 2012.
- [15] K. He et al., "Deep Residual Learning for Image Recognition," in Proc. CVPR, 2016.
- [16] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for CNNs," in Proc. ICML, 2019.
- [17] S. Han et al., "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," in Proc. ICLR, 2016.
- [18] B. Jacob et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in Proc. CVPR, 2018.
- [19] M. Sandler et al., "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in Proc. CVPR, 2018.
- [20] A. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017.
- [21] A. Geiger et al., "Vision Meets Robotics: The KITTI Dataset," IJRR, 2013.
- [22] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in Proc. ICLR, 2015.
- [23] Flask Documentation, "Flask Web Framework," 2023. [Online]. Available: <https://flask.palletsprojects.com>
- [24] T.-Y. Lin et al., "Microsoft COCO: Common Objects in Context," in Proc. ECCV, 2014.
- [25] Z. Cai and N. Vasconcelos, "Cascade RCNN: High Quality Object Detection," in Proc. CVPR, 2018.