big data and cognitive computing

Article

Real-Time Monitoring of Road Networks for Pavement Damage Detection Based on Preprocessing and Neural Networks

Nataliya Shakhovska [1,2,3], Vitaliy Yakovyna [4], Maksym Mysak [1], Stergios-Aristoteles Mitoulis [3,5], Sotirios Argyroudis [2,3] and Yuriy Syerov [6,7,*]

1 Artificial Intelligence Department, Lviv Polytechnic National University, 79013 Lviv, Ukraine; nataliya.b.shakhovska@lpnu.ua (N.S.); maksym.mysak.mknssh.2022@lpnu.ua (M.M.)
2 Department of Civil and Environmental Engineering, Brunel University of London, Uxbridge UB8 3PH, UK; sotirios.argyroudis@brunel.ac.uk
3 MetaInfrastructure.org, London NW11 7HQ, UK; s.a.mitoulis@bham.ac.uk
4 Faculty of Mathematics and Computer Science, University of Warmia and Mazury in Olsztyn, ul. Oczapowskiego 2, 10-719 Olsztyn, Poland; yakovyna@matman.uwm.edu.pl
5 Department of Civil Engineering, School of Engineering, University of Birmingham, Birmingham B15 2TT, UK
6 Social Communication and Information Activity Department, Lviv Polytechnic National University, 79013 Lviv, Ukraine
7 Department of Information Management and Business Systems, Comenius University Bratislava, 82005 Bratislava, Slovakia
* Correspondence: yurii.o.sierov@lpnu.ua

Abstract: This paper presents a novel multi-initialization model for recognizing road surface damage, e.g. potholes and cracks, on video using convolutional neural networks (CNNs) in real-time for fast damage recognition. The model is trained by the latest Road Damage Detection dataset, which includes four types of road damage. In addition, the CNN model is updated using pseudo-labeled images from semi-learned methods to improve the performance of the pavement damage detection technique. This study describes the use of the YOLO architecture and optimizes it according to the selected parameters, demonstrating high efficiency and accuracy. The results obtained can enhance the safety and efficiency of road pavement and, hence, its traffic quality and contribute to decision-making for the maintenance and restoration of road infrastructure.

Keywords: pavement; damage detection; convolutional neural network; YOLO architecture; machine learning; classification; neural networks; data preprocessing

check for updates

Citation: Shakhovska, N.; Yakovyna, V.; Mysak, M.; Mitoulis, S.-A.; Argyroudis, S.; Syerov, Y. Real-Time Monitoring of Road Networks for Pavement Damage Detection Based on Preprocessing and Neural Networks. *Big Data Cogn. Comput.* **2024**, *8*, 136. https://doi.org/10.3390/bdcc8100136

Academic Editors: Christine Dewi and Rung-Ching Chen

Received: 18 July 2024
Revised: 15 September 2024
Accepted: 8 October 2024
Published: 11 October 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The road surface is an important part of transport infrastructure and plays a key role in ensuring the safety and comfort of traffic. However, the road surface is subject to defects and damage over time due to heavy traffic, adverse weather conditions, and other factors. A damaged road surface, including cracks, holes, potholes, and other defects, causes discomfort for drivers and passengers and threatens road safety. Moreover, the restoration and repair of roads require significant financial costs and manpower. Fast and effective detection and classification of the damage type and severity is important for maintaining and restoring the road infrastructure and ensuring road safety toward infrastructure resilience [1].

One of the leading approaches to pavement damage detection today is deep learning, particularly convolutional neural networks (CNNs). CNNs have already proven to be highly effective in image and video processing tasks and can be a powerful tool for automated recognition of different types of pavement damage in video [2].

This work aims to develop a real-time model for recognizing road surface damage on video using CNN models for fast damage recognition. The information system developed based on the proposed model will be able to collect and proceed with multimodal information from various sources (e.g., social media, photos, and video). This will make it possible to quickly respond to malfunctions and carry out repairs on time, increasing the safety and extending the service life of road infrastructure.

The main objectives of this study include:

1. This study involves developing convolutional neural network (CNN) models capable of accurately and efficiently identifying road surface damage within video footage. The proposed approach entails modifying the YOLOv4-tiny model through the integration of data augmentation, hyperparameter optimization, utilization of a multi-initialization model, and implementation of a combined loss function. The primary objective is to attain a high level of accuracy and dependability in the identification of diverse forms of damage on road surfaces.
2. Real-time operation entails developing an algorithm and a system capable of analyzing video streams in real-time. This capability facilitates prompt damage detection and immediate response.
3. The system is designed to support and aid in the planning of road repair works. It can prioritize road surface restoration based on detected damage.

This paper is organized as follows: Section 2 highlights the benefits and drawbacks of various recognition models; Section 3 presents the proposed recognition model; Section 4 presents the results of training and testing the developed model; Section 5 compares the results obtained by different models such as YOLOv8, YOLOv9c, MobileNetSSD, and RTDERT for road pothole detection; Section 6 concludes the paper.

2. Related Work

The RDD2020 dataset [3] contains more than 26,000 road images from three countries and more than 31,000 cases of pavement damage, such as cracks and potholes. This dataset is intended for the development of deep learning methods for the automatic detection and classification of road damage. The images were captured using smartphones, which are essential for monitoring road conditions. This dataset can also be used in machine learning approaches to compare different algorithms and solve similar problems.

The pavement crack dataset, DeepCrack, VGG-16/UNet, and Resnet34/UNet model results for classification problem solving are presented in [4]. VGG-16/UNet achieves the highest accuracy, with an F1-score approximately equal to 0.646. This result highlights the importance of pavement damage detection and suggests methods for improvement.

Hacıefendioğlu K. and Başağa H. [5] deal with research in the field of road surface damage detection using image processing. The researchers make three important contributions to the problem of road damage detection. First, a large road damage dataset was prepared for the first time, including 9053 road damage images captured by a vehicle-mounted smartphone and containing 15,435 road damage cases. Next, they used advanced object detection techniques using convolutional neural networks to train a damage detection model using their dataset. They compared the accuracy and speed of running on a GPU server and a smartphone. Finally, the researchers demonstrate that damage types can be classified into eight types accurately using the proposed object detection method. The study results, the road damage dataset, and the smartphone application developed for this study are available for public use.

The authors from [6] raise the problem of road surface damage and their impact on road safety. The use of fully convolutional neural networks (CNNs) for detecting road surface damage using semi-learned learning methods is proposed. First, the training database is collected using a camera mounted on the car while driving on the road. The CNN model is trained in semantic segmentation using a deep convolutional autoencoder. The training dataset was augmented with luminance variations, and a total of 40,536 training images were generated. In addition, the CNN model is updated using pseudo-labeled images from semi-learned methods to improve the performance of the pavement damage detection technique. To test the effectiveness of the proposed method, 450 datasets were generated to evaluate pavement damage detection, and four experts evaluated each image. The results confirmed that the proposed method effectively determines the locations of road surface damage.

In [7], using deep learning to manage road infrastructure is considered to improve the safety and efficiency of the transport network. The study presents the You Only Look Once version 5 (YOLOv5) object detection model trained on the latest Road Damage Detection 2022 (RDD 2022) dataset, which includes four types of road damage. The YOLOv5 model has been improved using various techniques, including the Effective Channel Attention (ECA-Net) module, label smoothing, the K-means++ algorithm, focused loss, etc. As a result of these improvements, the accuracy and overall performance of the model have been improved compared to the base version of YOLOv5.

Article [8] considers the use of deep learning to solve real-world object detection tasks, in particular, road damage detection. The study describes using the YOLOv5x model to detect different types of pavement damage as part of the IEEE BigData Cup Challenge 2020. The YOLOv5x-based approach is characterized by ease and speed of operation while providing good detection accuracy. The study achieves an F1-score of 0.58 using an ensemble of image-augmented models, making it a potentially suitable candidate for real-time road damage detection [9].

The article's authors [10] investigate the need for an automated system using the 2020 IEEE Big Data Global Road Damage Detection Challenge dataset. Experimental results showed that the F1 index reached 0.67, which allowed the authors to win this competition.

In [11], the importance of detecting road damage for their effective maintenance, which was traditionally performed with the help of expensive high-performance sensors, is investigated. Thanks to recent advances in computer vision, detecting and classifying different types of road damage is now possible, contributing to efficient maintenance and resource management. This paper presents an ensemble model for efficient road damage detection and classification [12], presented at the IEEE BigData Cup 2020 challenge. The solution uses a state-of-the-art object detector called "You Only Look Once" (YOLO-v4), which was trained on images of various types of road damage from the Czech Republic, Japan, and India. The ensemble approach was extensively tested using several different model versions and achieved an F1 of up to 0.628 on the test 1 dataset and 0.6358 on the test 2 dataset.

The paper [13] overviews computer vision, machine learning, and structural engineering research. It analyzes two types of computer vision-based applications: inspection and monitoring.

The authors [14] conducted a study on monitoring forest roads, emphasizing the importance of cost-effective monitoring. They noted that smartphones are now being used to detect road surface degradation due to their advantages, availability, low cost, and expected accuracy. The study proposes a technological approach that uses YOLOv4-v5 to detect forest road damage in images and also extends the model's capabilities for orientation estimation tasks. The main goal is to provide quality information about the condition of forest roads and indications of possible damage.

Based on prior analysis, it is imperative to approach the primary task holistically for the automated detection of pavement damage, encompassing:

- Capturing high-quality data is crucial for training accurate and resilient recognition models. Such data can be obtained using mobile mapping systems and road cameras. Ensuring the reliability and consistency of data acquisition processes enhances the effectiveness of automated recognition systems.
- Feature extraction is crucial in identifying diverse forms of pavement damage, including cracks, potholes, rutting, and surface distress. It necessitates using practical algorithms to discern the unique characteristics of each type of damage by analyzing texture, color, shape, and spatial patterns present in pavement images.
- The utilization of classification algorithms in assessing pavement damage is a common practice, often involving the application of machine learning and deep learning

techniques. Supervised learning methods play a pivotal role in this context, as they entail training models on labeled data. Convolutional neural networks (CNNs) have exhibited substantial promise in pavement damage recognition, attributed to their capacity to autonomously acquire hierarchical features from unprocessed image data.

- Model training and validation for developing accurate recognition models necessitates the utilization of large and diverse datasets. Annotated datasets containing labeled examples of pavement damage are imperative for effectively training the models. Furthermore, stringent validation procedures are essential for evaluating the model's generalization ability across varied road conditions and environments.
- Real-time processing necessitates deploying automated recognition systems tailored for real-time applications, such as onboard vehicle systems or continuous monitoring networks. The efficacy of these systems hinges on the ability of algorithms to process substantial volumes of data in real-time efficiently. Therefore, optimizing the computational efficiency of recognition algorithms is paramount for their practical implementation.
- Pavement conditions differ greatly based on regions, climates, and traffic. Recognition models need to be adaptable and generalize well across diverse environments to be effective in real-world situations. Transfer learning techniques, which involve pre-training models on large datasets and fine-tuning them for specific target domains, can help improve adaptability.

Applying deep learning based on convolutional neural networks (CNN) is a promising approach for automated pavement damage recognition, which can facilitate the prioritization of restoration efforts and efficient allocation of resources. The main tasks include developing an accurate recognition system that effectively detects different types of damage in videos.

While deep learning and computer vision have shown significant promise in pavement damage detection, several drawbacks need to be addressed to enhance their practical applicability and effectiveness. Training complex models can be resource-intensive, requiring powerful hardware and extensive time. Models trained on specific datasets may not generalize well to different environments or conditions, such as varying road surfaces, lighting conditions, or weather patterns.

Overcoming data quality, computational requirements, generalization, real-time processing [15], robustness, integration, and cost challenges will be crucial for the widespread adoption and success of automated pavement damage recognition systems. Advanced models, such as those incorporating deep neural networks, require significant computational resources for training and real-time processing.

3. Materials and Methods

3.1. The Research Methodology

The research methodology consists of the following steps:

- In order to proceed, it is necessary to gather a substantial quantity of road surface videos from diverse sources, such as cameras or mobile devices, depicting various forms of damage, including potholes, cracks, and asphalt deterioration. Each video necessitates annotation to denote the precise location and nature of the damage. This is why preprocessing is a crucial component of this stage.
- The development of CNN architectures entails creating and optimizing deep neural network architectures to facilitate efficient video damage detection and classification. Additionally, it is imperative to rigorously assess the models' real-time analysis capabilities and capacity for processing large volumes of data.
- Model training and tuning encompass the utilization of forward and backward training methods. This process also entails refining model parameters to optimize accuracy.
- Evaluation and validation encompass testing developed models on test videos to assess their accuracy and reliability in damage detection. Subsequently, the obtained results are juxtaposed with those of existing methods.

– The integration and deployment phases encompass the development of an automated video damage recognition system and the integration-trained models. Implementing the MLOps paradigm is also undertaken to ensure the system's viability for real-time applications.
– This step involves evaluating the practical effectiveness of the developed system.

The flowchart of the proposed methodology is given in Figure 1.

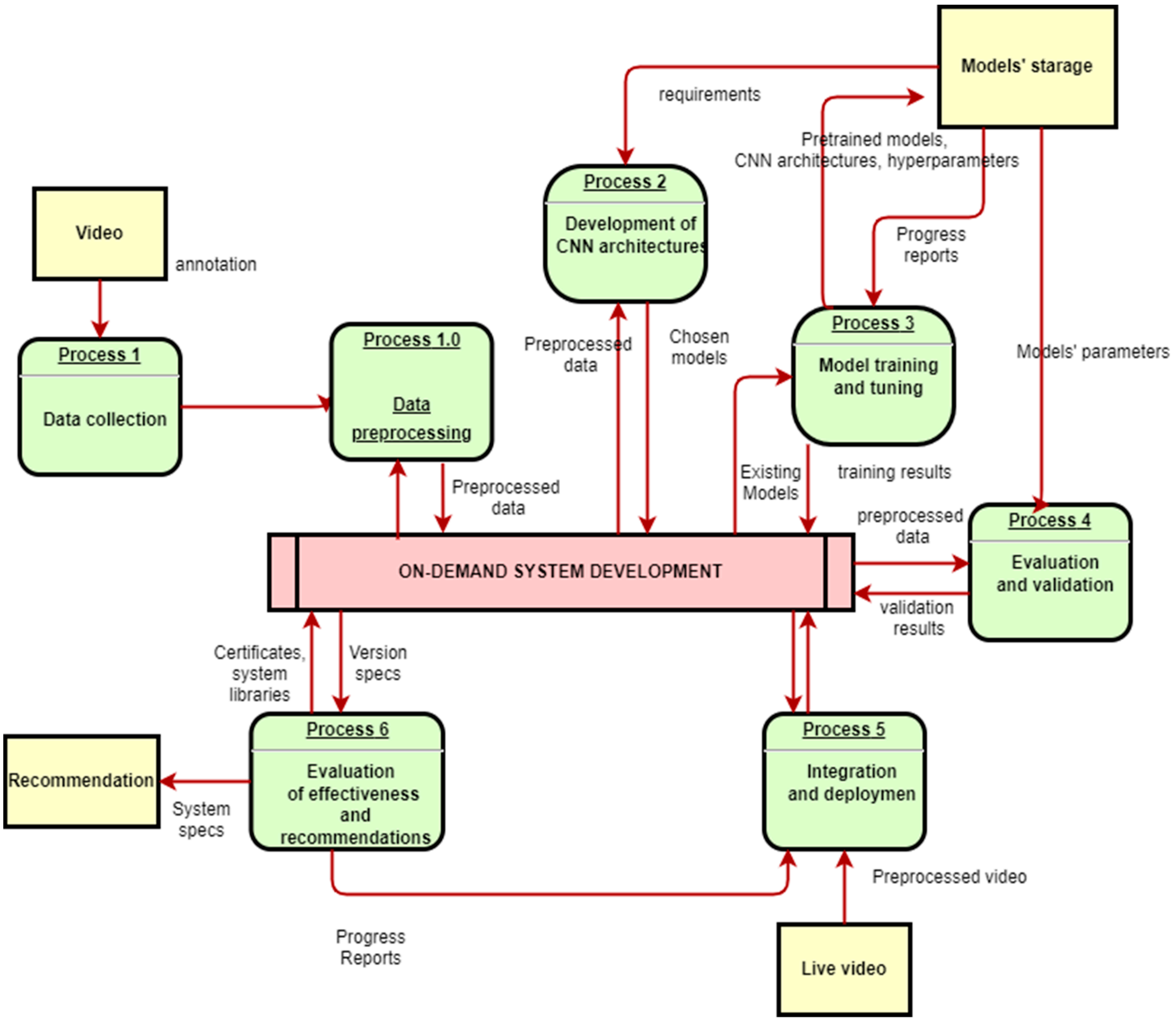


Figure 1. The flowchart of the proposed methodology.

The flowchart illustrates a comprehensive pipeline for developing and deploying CNN models, starting from data collection and preprocessing through model development, training, validation, integration, and ongoing evaluation. Each process is interconnected, ensuring a systematic approach to building a robust and effective machine-learning application. The following workflow is implemented: Video data are collected and annotated (Process 1). Data are preprocessed to prepare it for model training (Process 1.0). CNN architectures are developed based on the requirements and available data (Process 2). Models are trained and tuned using the preprocessed data (Process 3). Trained models are evaluated and validated to meet performance criteria (Process 4). Validated models are integrated and deployed to process live video streams (Process 5). The deployed system is evaluated for effectiveness, and recommendations are made for improvements (Process 6).

3.2. Dataset Description and Preparation

The dataset chosen for this study is a carefully curated and processed collection of over 5000 images of pavement defects captured in various conditions. This dataset [16] was created using a well-known data collection platform combining information from over 2000 locations. Each image in this set has been manually reviewed and verified by computer vision specialists from Datacluster Labs, ensuring the quality and reliability of the resulting data.

The dataset contains various images of pavement damage, including cracks (longitudinal, transverse, and alligator), potholes, rutting, and surface distress. These damage types have been selected based on their common occurrence and impact on road safety. The model has been specifically trained to identify and categorize these types of damage.

This dataset is perfect for training machine learning models to identify road defects. When discussing the dataset's features in the context of object detection, it is important to note that it is different from typical machine learning datasets. What sets it apart are the two labels for each object: one for classifying the object and the other for defining the bounding box that contains the object's position coordinates.

When we discuss approaches to model training, it is essential to note that datasets for object detection differ from standard machine learning datasets. Object detection datasets have two labels: one for the object's class label and another for each object's bounding box. The bounding boxes include the (x,y) coordinates of the four corners where the object is located. Several publicly available datasets for object detection tasks exist.

One critical aspect of this study is the absence of annotated datasets for pothole detection. Hence, it is vital to develop a dataset containing class labels and bounding boxes to specify the precise location of each image defect. Figure 2 presents an example of annotated data for training this model.

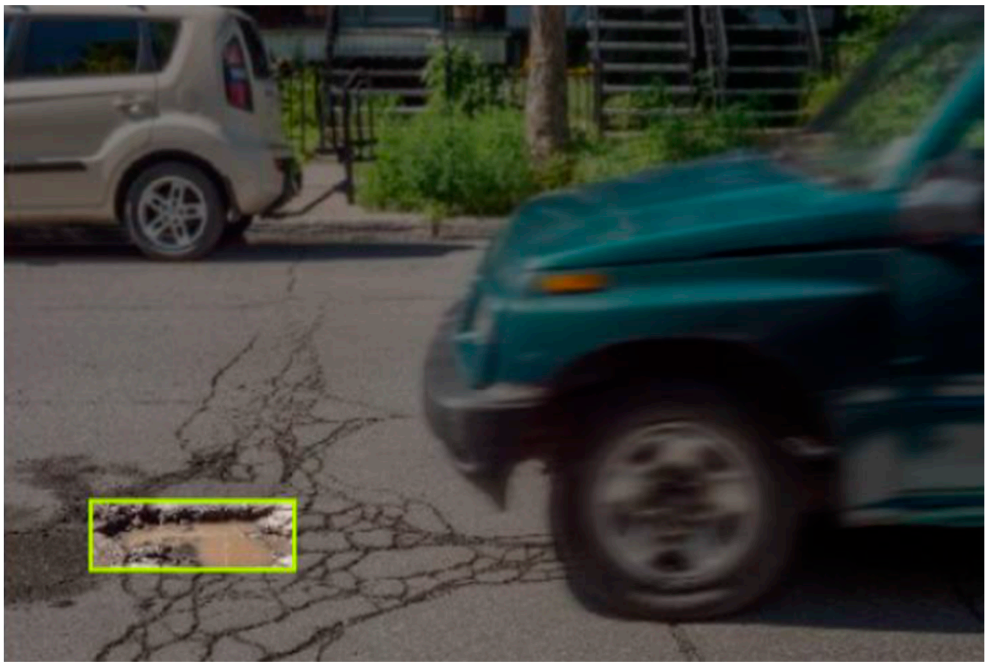


Figure 2. An example of annotated data for model training.

To ensure the most efficient training and evaluation of the model, the dataset was divided into three primary samples: training (70%), validation (20%), and testing (10%). This provides a balanced approach to model building and testing on various datasets, which is a crucial aspect of the successful implementation of this study.

3.3. Data Preprocessing

Grayscale images are often used in image processing because more minor data allow developers to perform more complex operations in a shorter time. There are several commonly used methods for converting a color image to a grayscale image, such as [17]:

1. Averaging method;
2. Weight method.

The averaging method calculates the average value of three parameters—Red, Green, and Blue (hereinafter RGB) as grayscale values:

$$I = \frac{R + G\_B}{3} \tag{1}$$

The averaging method is quite simple, but it does not work as well as we would like. Human eyeballs react differently to RGB. The eyes are most sensitive to green light, less sensitive to red light, and least sensitive to blue light. Therefore, the three colors should have different weights in the distribution. This brings us to the next point—the weight method.

The weight method weighs red, green, and blue colors according to their wavelength. The improved formula is as follows:

$$Grayscale = 0.299R + 0.587G + 0.114B \tag{2}$$

These weights in the weight method are not arbitrary. They are carefully chosen to reflect the typical human sensitivity to different colors. Green contributes the most to perceived brightness, followed by red and blue. This human-centric approach adds a layer of relatability to the technical process.

Image blurring and smoothing were also used for data preprocessing. Although it may seem counterintuitive, by reducing the image's detail, we can more easily find the objects of interest. In addition, it allows us to focus on larger structural objects in the images. In general, in practice, there are four main parameters of smoothing and blurring, which are often used in various types of projects [18]:

1. Simple medium blur;
2. Gaussian blur;
3. Median filter;
4. Two-way filtering.

Several advanced denoising filters have been developed recently, such as the Non-Local Means (NLM) filter and the Block-Matching and 3D filtering (BM3D) algorithm. These methods have demonstrated superior performance to traditional filters like the median filter in preserving image details while effectively reducing noise.

For example, the Non-Local Means (NLM) filter exploits the redundancy in natural images by averaging similar patches from different parts of the image. This leads to better noise reduction without sacrificing image details.

Similarly, the Block-Matching and 3D filtering (BM3D) algorithm leverages collaborative filtering and 3D transform domain processing to achieve impressive denoising results, particularly in challenging scenarios with high noise levels.

While the median filter may not always be considered state-of-the-art compared to techniques like NLM and BM3D, it still holds value in specific scenarios, and its usage can indeed be justified based on its characteristics and cost-effectiveness.

The median filter is relatively simple to implement and computationally efficient compared to more complex algorithms like NLM and BM3D. Due to its low computational cost, the median filter may be preferred in scenarios where computational resources are limited or real-time processing is required.

The median filter is robust to outliers and impulse noise. Unlike mean-based filters, which outliers can heavily influence, the median filter computes the neighborhood's median value, making it less susceptible to extreme noise values.

The median filter may be more cost-effective than sophisticated denoising algorithms regarding implementation and computational resources. This makes it a practical choice for applications where budget constraints or hardware limitations are a concern.

While the median filter may not always offer the highest denoising performance compared to advanced techniques, its simplicity, efficiency, robustness to outliers, and cost-effectiveness make it a viable option in various scenarios, mainly when real-time processing or resource constraints are considered. Therefore, depending on a given application's requirements and restrictions, the median filter can still be a valuable tool in image-denoising tasks.

This work uses the median filter method. A median filter is a nonlinear digital filtering technique often used to remove noise from an image or signal. Median filtering is widely used in digital image processing because, under certain conditions, it preserves edges while removing noise. This is one of the best noise removal algorithms.

A median filter passes through each signal element, in this case, an image, and replaces each pixel with the median of the neighboring pixels located in a square neighborhood around the pixel being evaluated. The only advantage of this method over Gaussian blur and frame blur is that in these two cases, the replaced center value may contain the value of a pixel that is not even in the image, making the image's color look different and strange. Still, in the case of a median blur, it will look much more natural since it takes the median of the values already present in the image.

The formula for determining the average brightness is given as

$$brightness = \frac{1}{N}\sum_{i=1}^{N} pixel_i, \tag{3}$$

where N is the number of pixels in a shade of gray frame.

The brightness and contrast correction formula is given as follows:

$$frame_{adjusted} = a * frame_{original} + \beta, \tag{4}$$

where a is the contrast ratio, β is the brightness ratio, $frameoriginal$ is the original video frame, $frameadjusted$ is the video frame after correction.

The brightness and contrast correction method adapts the input data to different lighting conditions, improving the visibility of objects in the video and facilitating the model's recognition of road surface damage.

3.4. Model Development and Procedure for Selecting the Optimal Operating Parameters

The procedure for selecting optimal parameters for deep learning models, particularly for convolutional neural networks (CNN), is crucial in developing effective object recognition systems in images or videos. Optimal parameters are an essential factor that determines the quality and performance of the model. The parameter selection process is often a resource-intensive task and many experiments. Determination of model parameters includes:

1. Choosing a CNN architecture. VGG, ResNet, Inception, and YOLO are used to fine-tune existing pre-trained models [19–22].
2. Size of input images. The images are 416 × 416 pixels, and they are used for model training and inference.
3. Learning rate.
4. Number of training epochs.
5. Batch size.
6. Loss function. YOLO models typically use a loss function to consider object constraints' recognition quality and accuracy.

7. Regularization, such as dropout or L1/L2 regularization, prevents the model from overtraining.
8. Optimizer. Adam and SGD (Stochastic Gradient Descent) are analyzed.
9. Detection threshold.

The architecture of the YOLO_tinyv4 model was chosen based on previous analysis of the accuracy of VGG16, ResNet, Inception, and YOLO [23–26] models for recognizing road surface damage. The batch size is 64, which indicates the number of images the model processes simultaneously. At the same time, only one is used

$$subdivision\ (subdivision = 1),$$

which means that the whole lot is processed in its entirety without additional division.

So, the preprocessing includes combining the median filter method and following data augmentation techniques: Random, Rotation, Saturation, Scaling, and Brightness using Formulas (3) and (4). Random rotations were applied within a range of ±15 degrees to simulate various camera angles. Images were randomly scaled by a factor of 0.8 to 1.2 to mimic variations in camera distance from the road surface. Horizontal flips were randomly applied to simulate changes in road orientation. Random translations along the x and y axes were used to simulate slight shifts in camera positioning.

The proposed architecture consists of 6 stages. The pre-processing stages, conducted before data input into the network, encompass standard data augmentation methods during the training phase. These techniques enhance the model's adaptability by simulating diverse real-world scenarios. The pre-processing stages guarantee that the input images are standardized and suitably adjusted, thereby priming them for optimal feature extraction by the network.

The network architecture comprises multiple convolutional layers. Each block comprises a convolution, a normalization layer, and a Leaky ReLU activation. Subsequent blocks entail convolutional layers, pooling, and concatenation layers. The initial block consists of 32 filters, a 3 × 3 kernel size, a stride of 2, padding of 1, and Leaky ReLU activation. The succeeding block entails 64 filters, a 3 × 3 kernel size, a stride of 2, padding of 1, and Leaky ReLU activation. The third block involves 64 filters, a 3 × 3 kernel size, a stride of 1, padding of 1, and Leaky ReLU activation. A "route" module combines the prior layer alongside multiple additional convolutional and pooling layers.

The algorithm utilizes a combination of convolution layers and routing layers to select the desired features within each block, thereby enabling the selection of different levels of abstraction for processing. It commences with the "route" module, which merges the previous layer with the grouped merged part, followed by convolutional layers featuring standard Leaky ReLU activations. Subsequently, the algorithm incorporates another "route" module to merge the previous layer with an additional layer and a "maxpool" module for pooling purposes.

Each YOLO block consists of convolutional layers designed for feature extraction, and an associated YOLO block is used for prediction.

The configuration of parameters in object detection plays a crucial role in enhancing accuracy. These parameters include anchors, which determine the search areas for objects, and classes, defining the number of object categories the model should recognize. Additionally, these parameters are integral to the calculation of the loss function.

The Rectified Linear Unit (ReLU) activation function allows large negative values to be passed and only activates positive values, thereby mitigating the "vanishing gradient" problem.

The configuration also includes parameters for training, such as learning_rate, batch_normalize, filters, and others, which regulate the process of optimizing and training the model. The proposed model architecture is given in Figure 3.

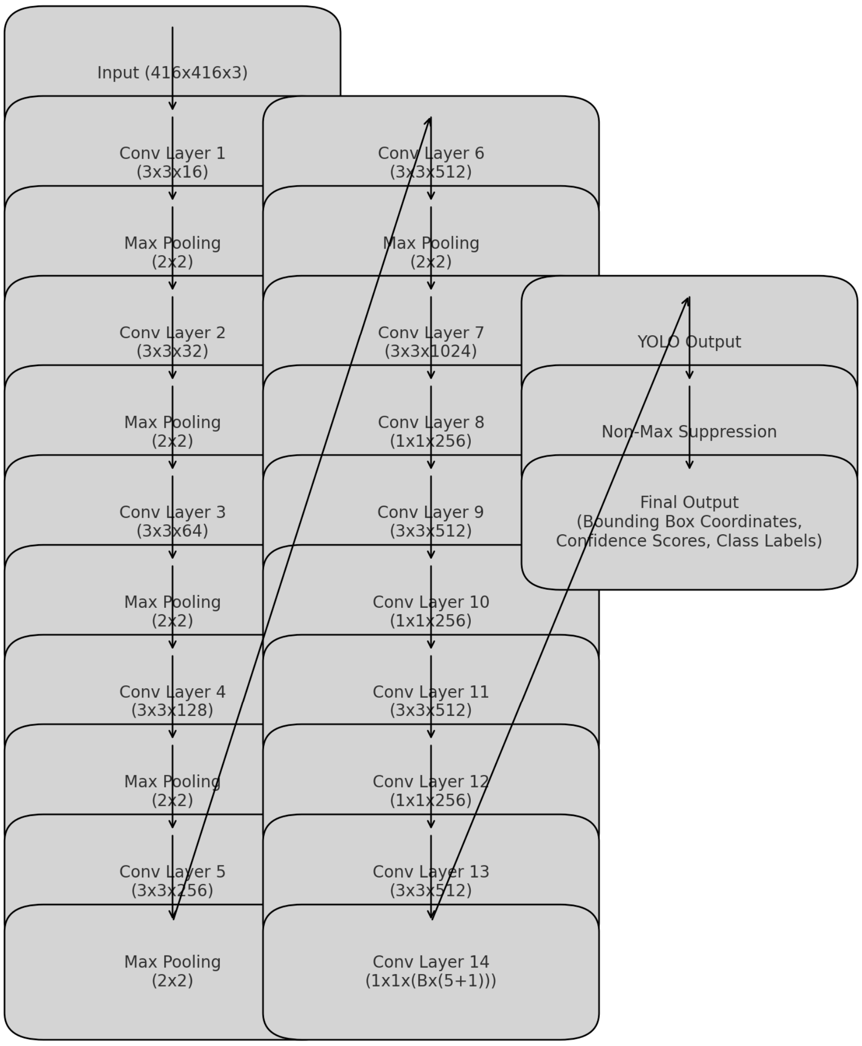


Figure 3. Proposed model architecture diagram.

The dimensions of the input image for the model are set to 416 $\times$ 416 pixels (width = 416, height = 416), which helps to maintain proportions during object recognition. The number of channels of the input image is left standard and equal to 3 (channels = 3), corresponding to the RGB format.

Optimization hyperparameters include momentum factor (momentum = 0.9) and decay factor (decay = 0.0005). These values help maintain the stability of the training process and reduce the risk of overtraining the model.

Image augmentation uses parameters such as rotation angle (angle = 0), saturation (saturation = 1.5), brightness (exposure = 1.5), and hue (hue = 0.1). These settings allow the model to learn under different illumination conditions and ensure the algorithm's operation in various contexts. The learning rate is set to 0.00261, and the number of first iterations with a lower learning rate (burn_in) is 1000. The maximum number of iterations for training (max_batches) is 2,000,200. As for the learning rate change strategy, the steps policy is used, with rate change steps of 1,600,000 and 1,800,000 iterations. The learning rate scales at these steps are set to 0.1 for both cases.

This parameter tuning approach enables efficient training of the YOLO_tinyv4 model for road pothole recognition, ensuring optimal accuracy and stability under various road environment conditions.

The modification of the YOLOv4-tiny model for pavement damage detection is made. In the first stage, augmentation techniques like random, rotation, saturation, scaling, and brightness are used to improve model generalization. Next, model optimization with hyperparameter tuning and loss function adjustment is implemented. To effectively train an object detection model, loss of intersection and union and classification loss are combined into a single loss function. A typical loss function for YOLOv4-tiny combines the IoU loss for bounding box regression, the classification loss for object classification, and possibly additional losses for objectness score and other factors. We propose to take into account additional boundary-aware penalties to improve the localization accuracy of detected potholes:

$$\begin{gathered} \textit{Total Loss} = \lambda_{coord} \cdot \textit{IoU Loss} + \lambda_{class} \cdot \textit{Classification Loss} + \lambda_{obj} \cdot \textit{Objectness Loss} \\ + \lambda_{boundaryj} \cdot \textit{BoundaryPenalty}, \end{gathered} \tag{5}$$

where λ_{coord} is a weight for the IoU loss (bounding box regression), λ_{class} is a weight for the classification loss, λ_{obj} is a weight for the objectness loss, which measures the confidence that an object exists in the predicted bounding box, and $\lambda_{\text{boundaryj}} \cdot$ presents the distance between the edges of the predicted and ground truth bounding boxes using the L1 norm (sum of absolute differences).

Next, the outputs of multiple YOLOv4-tiny models trained with different initializations are combined to improve robustness and accuracy. The model was continuously updated with new real-time data to keep it relevant and accurate.

3.5. Performance Evaluation Indicators of Models

Evaluating the performance of machine learning models, including CNN models, is essential to determining their performance and suitability for a particular task. There are several key indicators for evaluating the effectiveness of the model [27,28]:

- Accuracy measures the percentage of correctly classified examples relative to the total number of examples in the dataset (*tp*—true positive, *tn*—true negative, *fp*—false positive, *fn*—false negative):

$$A = \frac{tp + tn}{tp + tn + fp + fn}.$$

- Precision defines the percentage of correctly classified positive examples relative to all examples that the model recognized as positive. It is crucial to avoid false positive results in cases where

$$P = \frac{tp}{tp + fp}.$$

- Recall measures the percentage of correctly classified positive examples relative to all actual positive examples in the dataset.

$$R = \frac{tp}{tp + fn}.$$

- F1 score combines accuracy and completeness into one metric and measures the balance between the two. It is instrumental in situations where accuracy and completeness are weighted differently.

$$F1 \frac{2 \times (P \times R)}{P + R}.$$

- ROC curve displays the relationship between the percentage of true positives and the percentage of false positives at different classification thresholds. AUC measures the area under the ROC curve and indicates the overall ability of the model to distinguish classes.

- Mean Average Precision (MAP) is used to evaluate models that use ranking results, such as object retrieval tasks. It determines the average value of accuracy at different thresholds:

$$maP = \frac{1}{N}\sum_{i=1}^{N} AP_i,$$

where AP_i is the Average Precision for each class, and N is the number of classes.
- Mean IoU is used for object segmentation tasks. Measures the mean value of the union of intervals between predicted and actual feature segments.

Evaluating model performance requires careful metrics analysis and comparisons with a baseline or other models to determine how well the model solves the task.

4. Results

The developed model was trained on a trained dataset and on real video. The necessary tool list is given below:

- PyTorch;
- OpenCV;
- NumPy;
- Matplotlib;
- Pillow;
- TQDM;
- Nvidia TITAN X (Pascal).

The computer's operating system is based on the Windows 10 platform. Python is the programming language, while the integrated development environment utilized is the VS Code notebook, Anaconda. This will facilitate deployment and visualization [29].

Figure 4 shows iou_loss (loss of intersection and union), which determines the overlap between the predicted and actual regions. In this case, iou_loss is equal to 0.000000, which indicates a high accuracy of matching regions.

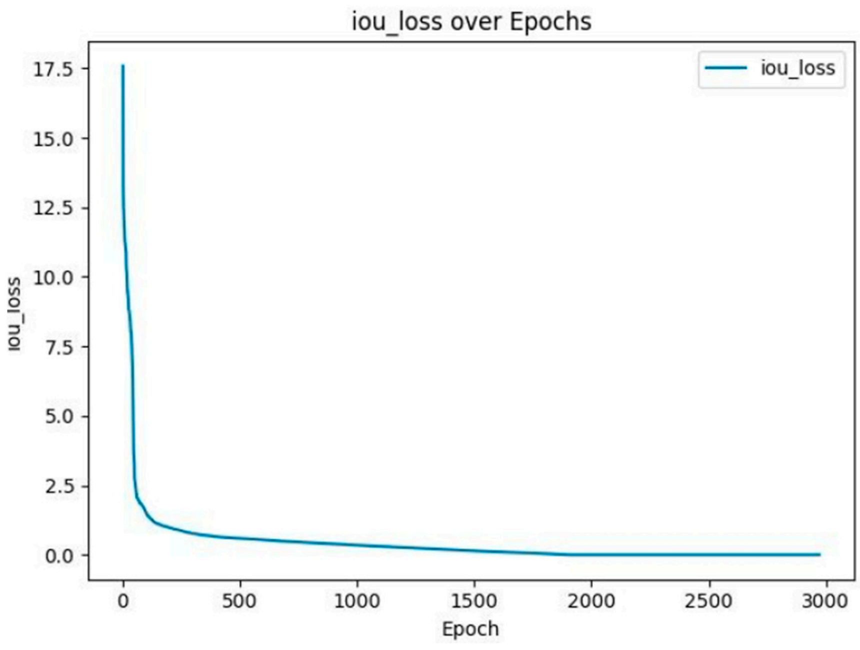


Figure 4. Visual presentation of the iou_loss metric.

Figure 5 shows the total loss, calculated as the sum of iou_loss and class_loss, is 0.000008.

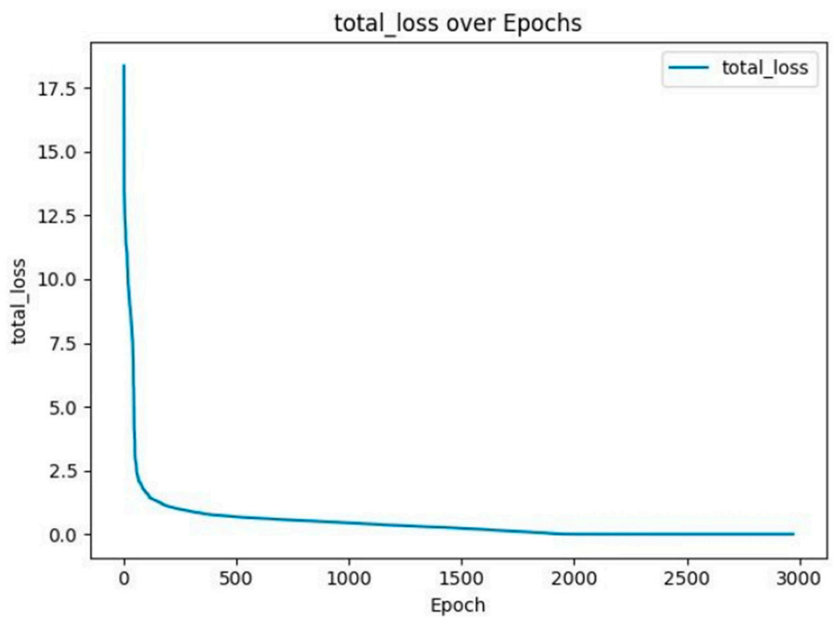


Figure 5. Visual presentation of the total loss metric.

The analysis begins with an assessment of the adaptation of the model to the definition of areas and classes of objects. According to the results, the value of total_bbox (the number of areas processed) is 432,640. This indicates the model's exceptional performance in recognizing objects in input images, instilling confidence in its capabilities. It should also be noted that only 0.000925% of regions were rewritten (rewritten_bbox). This indicates a high training stability of the model, meaning that the model has effectively adapted to the training data and exhibits few errors or confusion.

Figure 6 shows the class_loss metric, which measures object classification accuracy. The value of this metric is 0.000008, indicating a high accuracy of object class determination. This reassures the team about the model's reliability in recognizing and classifying objects in images.

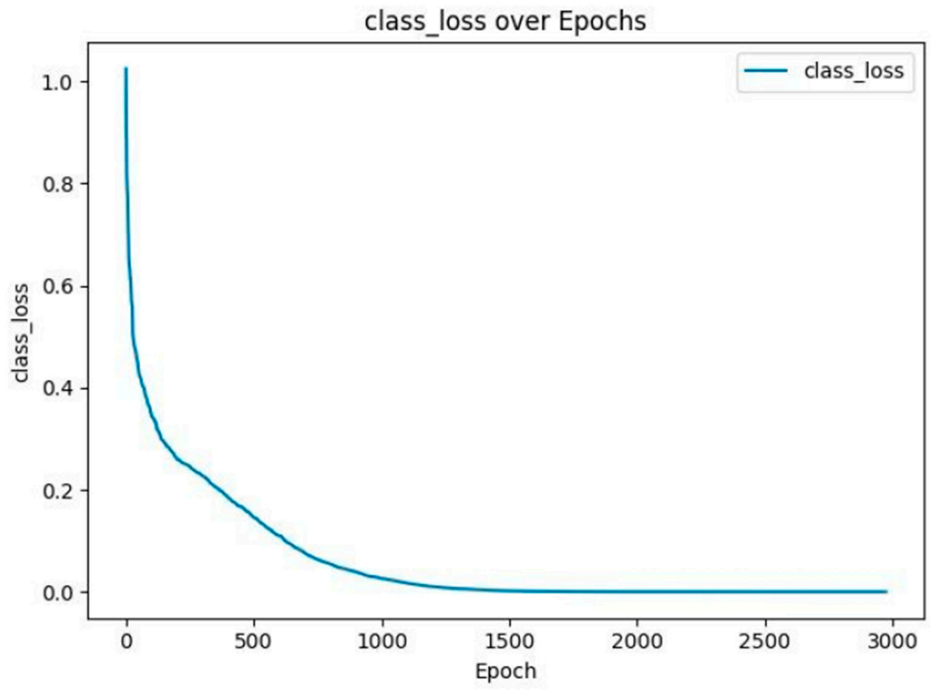
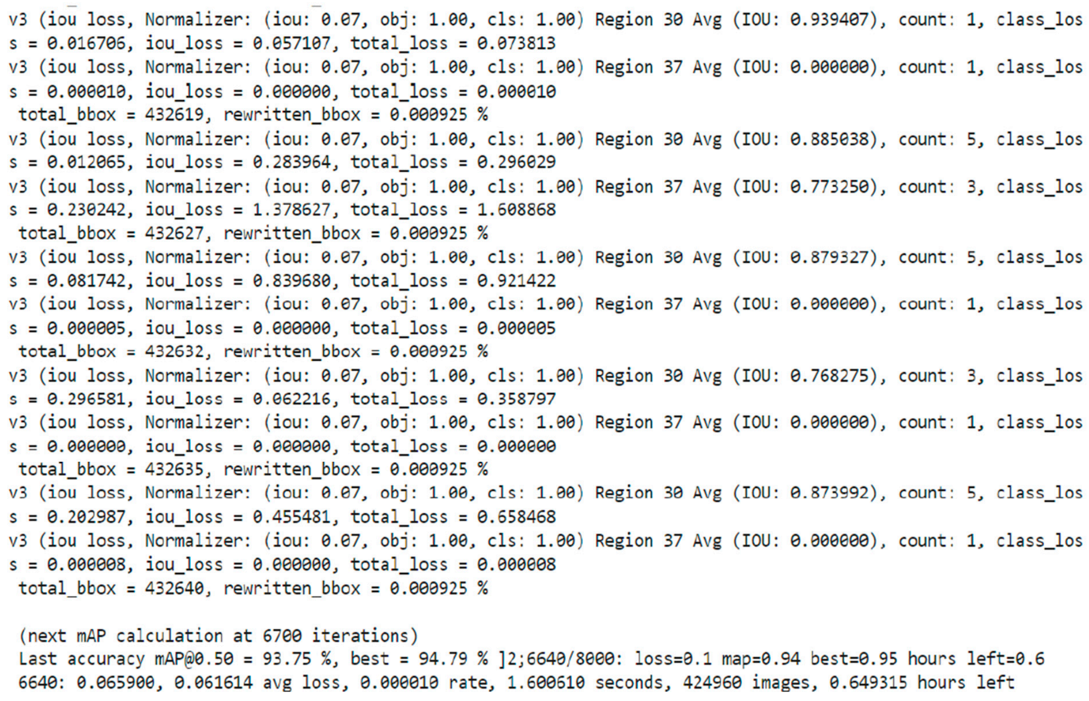


Figure 6. Visual presentation of the class_loss metric.

The training was conducted on images of size 416 × 416 and batches of size 64, with a learning rate of 0.00261 and other optimal parameters. These parameters led to training success, as reflected in the above metrics. However, it is essential to note that further adjustments are possible to achieve even higher accuracy and overall efficiency of the model. This is particularly crucial given the task's specifics, which may include a variety of conditions and variations.

Thus, further analysis and parameter adjustment can contribute to an even better adaptation of the model to variable testing conditions. The model training results for the last iterations are given in Figure 7.



```
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.939407), count: 1, class_los
s = 0.016706, iou_loss = 0.057107, total_loss = 0.073813
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_los
s = 0.000010, iou_loss = 0.000000, total_loss = 0.000010
 total_bbox = 432619, rewritten_bbox = 0.000925 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.885038), count: 5, class_los
s = 0.012065, iou_loss = 0.283964, total_loss = 0.296029
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.773250), count: 3, class_los
s = 0.230242, iou_loss = 1.378627, total_loss = 1.608868
 total_bbox = 432627, rewritten_bbox = 0.000925 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.879327), count: 5, class_los
s = 0.081742, iou_loss = 0.839680, total_loss = 0.921422
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_los
s = 0.000005, iou_loss = 0.000000, total_loss = 0.000005
 total_bbox = 432632, rewritten_bbox = 0.000925 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.768275), count: 3, class_los
s = 0.296581, iou_loss = 0.062216, total_loss = 0.358797
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_los
s = 0.000000, iou_loss = 0.000000, total_loss = 0.000000
 total_bbox = 432635, rewritten_bbox = 0.000925 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.873992), count: 5, class_los
s = 0.202987, iou_loss = 0.455481, total_loss = 0.658468
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_los
s = 0.000008, iou_loss = 0.000000, total_loss = 0.000008
 total_bbox = 432640, rewritten_bbox = 0.000925 %

 (next mAP calculation at 6700 iterations)
 Last accuracy mAP@0.50 = 93.75 %, best = 94.79 % ]2;6640/8000: loss=0.1 map=0.94 best=0.95 hours left=0.6
 6640: 0.065900, 0.061614 avg loss, 0.000010 rate, 1.600610 seconds, 424960 images, 0.649315 hours left
```

Figure 7. Results of model training on the last iterations.

As seen from the figure, the model's accuracy rate is presented. The training time for the proposed model was 28,144.66 s vs. 23,778.54 for the original YOLOv4-tiny model, which is higher due to the inclusion of the data preprocessing stage. The last value is mAP@0.50, equal to 93.75%. This indicator points to a high degree of reliability of object recognition in images when using a threshold of 0.50. Such a high level of accuracy should instill confidence in the team about the model's performance in real conditions, particularly for damage detection on the road surface. The last iteration's metrics show the overall stability of the training process. A consistent and low loss value suggests the model's stable training and good generalization capabilities. The total number of images processed is 424,596. 6640 iterations out of a total of 8000 have been completed.

Loss values were considered average and individual losses for each iteration. Average losses at each iteration are 0.065900, demonstrating the training stability and the absence of significant deviations. The total average losses during training are 0.061614, and the validation loss is 0.061721, indicating a good balance of the model and its ability to avoid overtraining or undertraining.

Calculating the learning rate shows a dynamic strategy for increasing the learning rate during training. This can be key to ensuring efficient model training and high accuracy rates. The highest achieved accuracy is 94.79%, which indicates the model's great potential in detecting and classifying objects in images.

In addition to the model's successful convergence, the analysis of training iterations provides valuable insights into the optimization process and the model's performance. By passing 6640 out of the planned 8000 iterations, the model demonstrates robustness and resilience throughout the training phase. This indicates that the model effectively learned from the training data and consistently progressed toward convergence.

Adjusting the learning rate every 1000 iterations was crucial in ensuring stability and accuracy during training. This adaptive learning rate strategy helps the model navigate complex optimization landscapes more effectively, allowing it to converge to a more optimal solution. By periodically reducing the learning rate, the model can fine-tune its parameters and refine its predictions, ultimately improving performance.

The duration of each iteration, averaging approximately 1.600610 s, provides valuable insights into the computational efficiency of the training process. This metric is essential for understanding the practical feasibility of deploying the model in real-world scenarios where efficient processing is critical. The relatively short duration of each iteration indicates that the training process is not overly burdensome regarding computational resources, further affirming the model's suitability for deployment in resource-constrained environments.

With only 0.649315 h remaining until the end of training, the model is on track to complete its optimization process within a reasonable timeframe. This remaining time estimate offers valuable planning information and allows stakeholders to anticipate when the trained model will be ready for deployment or further evaluation.

In summary, the successful convergence of the YOLOv4-tiny model, coupled with the adaptive learning rate strategy, efficient iteration duration, and remaining training time estimate, collectively affirm the model's readiness for deployment and its potential effectiveness in real-world applications such as road damage detection.

During the testing of the developed model, significant difficulties were found in recognizing damage to the road surface under conditions of insufficient lighting.

In particular, the model's parameters need to be optimized to improve its sensitivity to limited lighting conditions. Further expansion of the dataset includes increased.

As part of the model's optimization, it was decided to focus on improving the performance and quality of video data processing. This stage includes an important process of video data preprocessing, which is determined to achieve optimal results. Figure 8 shows the input data in the form of a frame from a video stream.

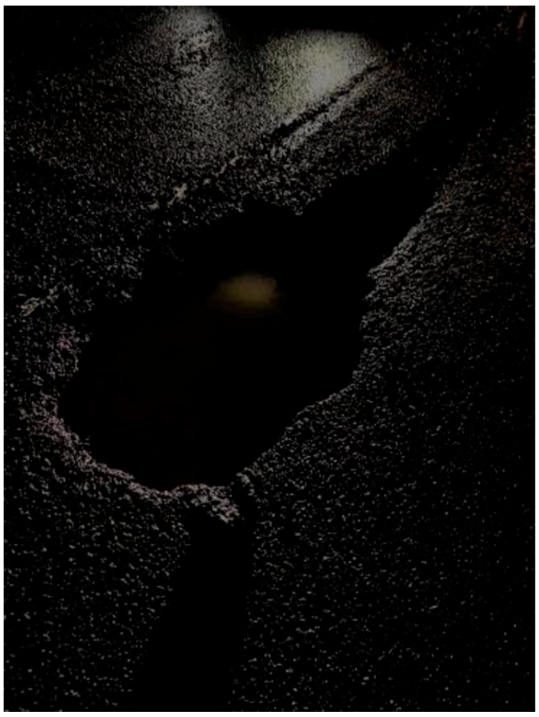


Figure 8. Input image (without preprocessing).

The first step in solving the problem is to analyze the video's brightness [30–32]. For this purpose, an algorithm converts each video frame to shades of gray and calculates the average brightness value. The main steps of the algorithm are given below.

1. **Load the Video**. Open the video file and initialize a loop to read each frame.
2. **Convert Frame to Grayscale**. For each frame, convert it to grayscale gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY).
3. **Calculate Average Brightness**. Compute the average brightness (or intensity) value of the grayscale frame average_brightness = gray_frame.mean().
4. **Store/Display Results**. Store or display the average brightness value for further analysis or visualization.

This approach makes it possible to obtain representative data on the lighting of the video material. Next, dynamic correction is applied based on the specified brightness value. Using algorithms for converting frames to shades of gray and functions of the OpenCV library, the contrast and brightness of video frames are changed based on Formula (1). This process is mathematically described through appropriate functions and algorithms, which allows us to achieve the balance of shades of gray and improve video data's visual quality.

The application of brightness and contrast correction allows the adaptation of input data to different lighting conditions, improving the visibility of objects in the video and facilitating the work of the model in recognizing road damage coating. An example of a frame with detected damage is shown in Figure 9. Analysis of the brightness of the input video allows us to determine the need for correction and apply it only if necessary, which reduces computational costs.

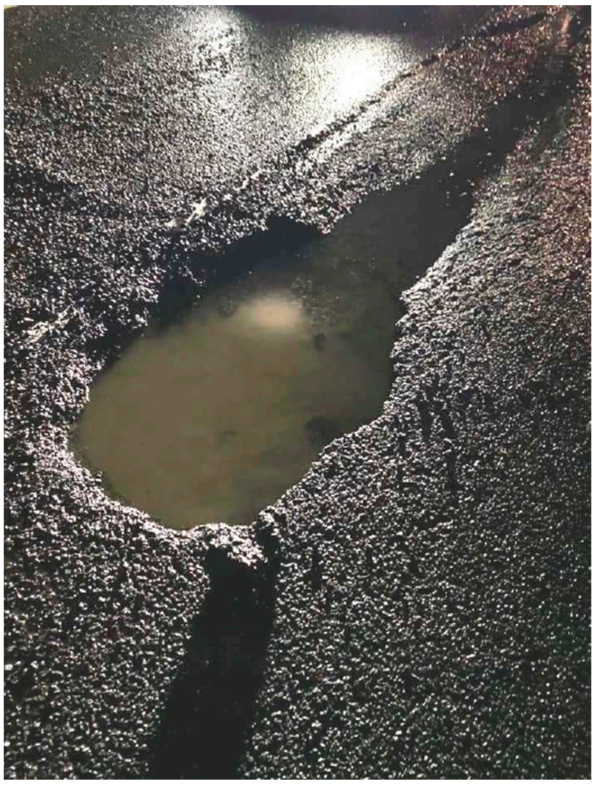


Figure 9. An example of an image after preprocessing.

The developed program could be used in real-time analysis. To do that, pseudo-labeled training is used. After training the initial CNN model on the labeled dataset, pseudo-labeled images are generated by applying the trained model to unlabeled data. The model's predictions on the unlabeled data are then used as "pseudo-labels" for these images. Each video frame is fed into the trained CNN model, which generates predictions for pavement damage detection. These predictions include bounding boxes indicating the presence of damage. The predictions generated by the model on each frame are treated as pseudo-labels for that frame. These pseudo-labels indicate where the model believes pavement damage is present in the video frames. The original labeled dataset, consisting of manually annotated frames, is augmented with the pseudo-labeled frames generated from the video. This combined dataset, containing labeled and pseudo-labeled frames, is then used to retrain the CNN model. The model is retrained using the augmented dataset, incorporating the ground truth labels from the original dataset and the pseudo-labels from the video frames. This allows the model to learn from both data types and improve its performance on pavement damage detection tasks.

Figures 10 and 11 demonstrate the detection of multiple potholes.

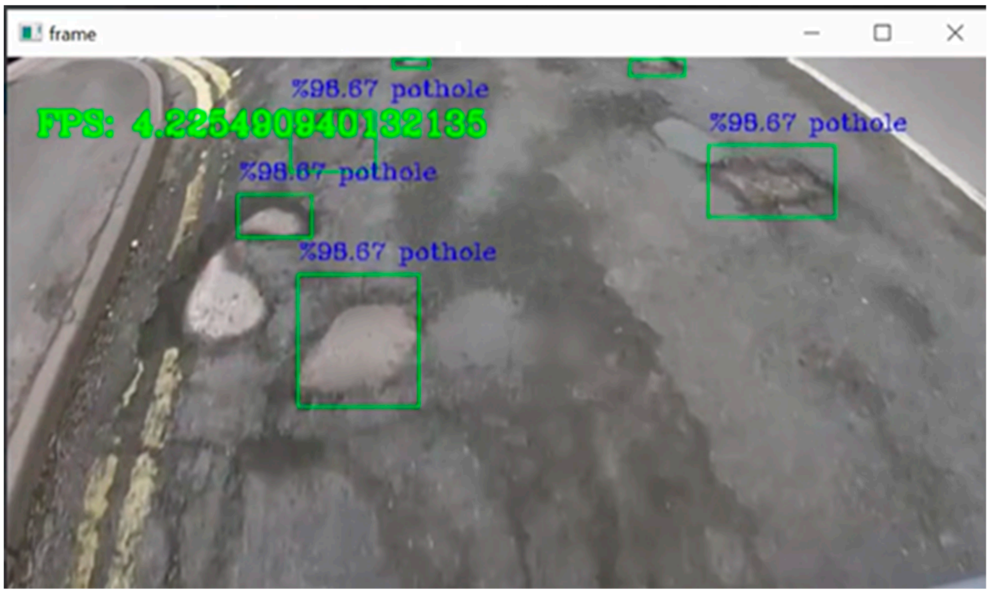


Figure 10. An example of multiple damage detection in real-time.

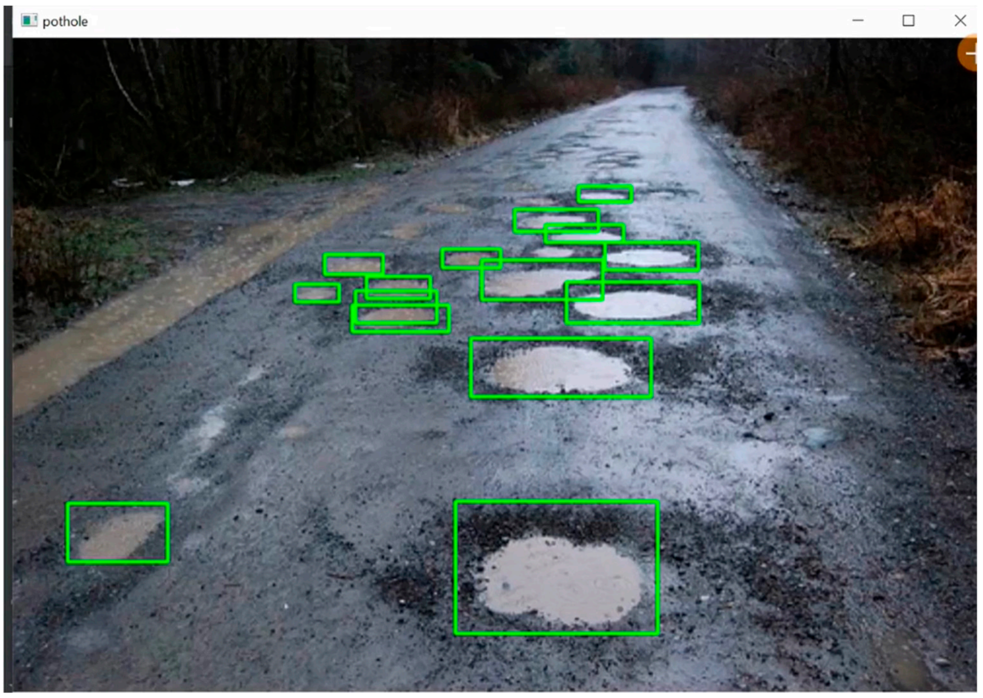


Figure 11. An example of multiple potholes on the road after rain in real-time.

Our model's architectural foundation is rooted in the standard YOLOv4-tiny framework. However, it has undergone several optimizations tailored to augment its performance for real-time applications:

- We have optimized the input resolution by fine-tuning it to balance speed and accuracy. This ensures that the model efficiently processes images while maintaining high detection performance.
- To minimize computational overhead, the lightweight preprocessing pipeline incorporates carefully selected steps, such as grayscale conversion and contrast enhancement. These measures expedite processing times without compromising the quality of detection.
- Data augmentation was exclusively utilized during training to maintain an efficient and streamlined inference phase.

To validate our real-time performance claims, we conducted benchmark tests comparing the processing times of our model with a standard baseline. The baseline is defined as the unmodified YOLOv4-tiny architecture without the abovementioned optimizations. The tests were performed using an NVIDIA GTX 1650 GPU, and the results are as follows:

Proposed Model:

- Inference Time per Image, 5.6 milliseconds
- Frames per Second (FPS) during Video Stream Processing, 178 FPS

Baseline Model (Standard YOLOv4-tiny):

- Inference Time per Image, 6.3 milliseconds
- Frames per Second (FPS) during Video Stream Processing, 159 FPS

The optimized model demonstrates a 12.5% improvement in inference time per image and an 11.9% increase in frames per second (FPS) compared to the baseline. This improvement is crucial for real-time applications, where even a millisecond can make a difference. The reduced processing time enables our system to process video streams at a speed that comfortably exceeds the real-time threshold, allowing for deployment in more complex environments or on less powerful hardware.

5. Discussion

The results obtained can potentially improve road pavement safety and efficiency and, hence, traffic quality and contribute to decision-making for road infrastructure maintenance and restoration. Video (sequence of images) and images from road cameras, mobile phones, or UAV-captured images could be processed [33].

Data analysis and preprocessing results indicate the importance of input data preparation to achieve optimal model outputs. The data analysis, preliminary processing, and augmentation improved the quality of learning and the model's generalization ability. The model's input is images or video, and the output is the localization of the damage. Interpreting the model's results provides an essential perspective on understanding and using the preprocessed images. The knowledge obtained allows us to determine the model's strengths and weaknesses and develop strategies to improve its work further.

The comparison of the proposed model with existing solutions is given in Table 1.

YOLOv4-tiny has a high mAP@0.50 of 93.75%, fast inference speed, and low computational requirements, making it highly suitable for real-time applications, such as on-vehicle deployment for road inspection. This model might not capture very small or subtle damages as accurately as other models like Faster R-CNN, which excels in detailed object detection. On the other hand, improved YOLOv4-tiny with a preprocessing stage allows increasing suitability for real-time use.

Real-Time Detection Transformer (RT-DETR, https://docs.ultralytics.com/models/rtdetr/, accessed on 12 November 2023) is a transformer-based model designed for object detection. It is known for its ability to model long-range dependencies and complex spatial relationships. However, it can be computationally intensive and may require much data and computational power to perform optimally.

Table 1. The comparative analysis of the results.

Methods	Params (M)	Precision	Recall	mAP@0.50
YOLOv8 Nano	6.3	0.8471	0.8528	0.6849
YOLOv8 Small	22.5	0.8608	0.854	0.7139
YOLOv8 Medium	52	0.873	0.855	0.8318
MobileNetv2SSD	8.48	-	0.8337	0.8881
RTDERT	66.5	0.674	0.769	0.772
YOLOv9c	51	0.773	0.854	0.7
YOLOv4-tiny with preprocessing stage and modified loss function (developed)	52	0.895	0.8591	0.9375
YOLOv4-tiny [26]	42	0.873	0.8591	0.92
YOLOv5 [9]	49	0.783	0.861	0.71
Faster R-CNN [5]	25.9	0.854	0.891	0.9

On the other hand, YOLOv4-tiny is more likely to perform better with smaller datasets or less complex scenarios. It is specifically designed to be efficient and requires fewer training resources. In specific contexts, DETR may underperform if it does not have access to large-scale data or sufficient training epochs, leading to poorer results than YOLOv4-tiny. The simplicity, efficiency, and targeted optimization of YOLOv4-tiny provide an edge in this context, particularly regarding inference speed and ease of training. In contrast, DETR's strengths in handling complex scenarios [34] may not translate into better performance for this specific application, leading to the observed discrepancies in mAP scores. This explains the worse result of RT-DETR than a developed model of YOLOv4-tiny with the preprocessing stage and modified loss function.

Faster R-CNNs are highly accurate in detecting small, intricate details and robust to various pavement damage types. However, they have a slower inference speed and higher computational load, making them less suitable for real-time applications.

The best precision is 0.89 for YOLOv4-tiny with the preprocessing stage and modified loss function, and the best recall is for Faster R-CNN and equals 0.891. The best mAP@0.50 is for YOLOv4-tiny with the preprocessing stage and modified loss function (0.9375). The YOLOv4-tiny, with the preprocessing stage and modified loss function, stands out as the best overall performer in precision and mAP, likely due to the added preprocessing stage and modified loss function. This indicates that these modifications significantly enhance the model's ability to accurately detect and classify objects (potholes in this context).

The results highlight the trade-offs between model complexity (number of parameters) and performance (precision, recall, mAP). More extensive models like YOLOv8 Medium and the modified YOLOv4-tiny tend to perform better, but simpler models can still achieve competitive results with appropriate modifications.

The YOLOv4-tiny model itself is designed for high efficiency, and with our optimizations, the additional preprocessing step does not hinder the system's real-time capabilities. It enhances the robustness and reliability of detections, leading to a more efficient overall system.

YOLOv4-tiny is known for its efficiency and speed, mainly when dealing with minor or specific types of objects. Our dataset includes small or distinct road damages (e.g., potholes, cracks). YOLOv4-tiny is better suited for detecting these features due to its ability to handle smaller input sizes effectively, leading to higher mAP scores. YOLOv4-tiny is a streamlined version of the entire YOLOv4 architecture, designed for faster inference with lower computational requirements. This simplicity could allow the model to perform better on specific tasks where the complexity of larger models (like YOLOv8) leads to overfitting or unnecessary processing overhead, which does not translate to better performance on your dataset.

The training data we used in our study were better suited for YOLOv4-tiny, as it consisted of more minor, well-annotated datasets with consistent object types. This likely contributed to higher mAP scores. On the other hand, YOLOv8, being a more complex model, may require more diverse and extensive datasets to utilize its capabilities fully. Additionally, our augmentation techniques were more compatible with the simpler YOLOv4-tiny architecture. For example, if the augmentations emphasized features that YOLOv4-tiny is better at detecting, such as simple rotations and scaling, this could disproportionately improve its performance compared to YOLOv8.

It is possible that YOLOv4-tiny could have performed better if we had fine-tuned the hyperparameters more effectively for our specific task. This would have led to better training convergence. Learning rate, batch size, and augmentation strategies could have been better optimized for YOLOv4-tiny. On the other hand, YOLOv8, being more complex, might be more susceptible to overfitting when trained on a smaller or less diverse dataset. This could result in YOLOv8 not generalizing well, leading to a lower mAP score in a specific testing environment than YOLOv4-tiny.

In summary, YOLOv4-tiny has achieved a higher mean Average Precision (mAP) in our specific case due to its compatibility with the dataset's characteristics, more effective training augmentations, better generalization on simpler models, and efficiency in processing small and distinct objects such as road damages.

Improved YOLOv4-tiny demonstrates a promising balance of accuracy, speed, and computational efficiency, making it an excellent choice for real-time pavement damage detection. However, models like Faster R-CNN may still be preferred for applications requiring the detection of outstanding details despite their higher computational costs.

The deployment of the model in practical settings requires a comprehensive approach that considers the unique demands of each environment. Whether deployed in a centralized system for broad coverage or in distributed edge devices for localized processing, the model's scalability ensures that it can meet the diverse needs of different users. Furthermore, the model can be integrated into large-scale, real-time monitoring systems by utilizing cloud-based infrastructure. This provides valuable insights into road conditions and enables proactive maintenance strategies.

A cloud-based deployment strategy enables the model to efficiently scale and process large amounts of road damage data gathered from different sources across a network. In cloud architecture, the model can take advantage of the elasticity of cloud resources to easily adjust based on demand, ensuring that processing power matches the current data processing requirements. This is especially beneficial for large-scale implementations, such as state or national road monitoring systems, where data from multiple regions need to be processed and analyzed simultaneously.

6. Conclusions

This paper is a first of its kind in recognizing road surface damage, e.g. potholes and cracks, using a multi-initialization model for on video using convolutional neural networks (CNNs) in real-time for fast damage recognition. The model was trained with the latest Road Damage Detection datasets. A description of the procedure for selecting the optimal parameters of the models was carried out, where the process of choosing hyperparameters that determine the operation of the models was described in detail. The selection of optimal parameters is an essential task in achieving the highest accuracy and efficiency of the system. The model performance metrics provide an objective overview of key metrics that measure model performance. The highest accuracy was achieved, which was 94.79%, IoU = 0.87, and mAP = 0.94. These metrics were used for the comprehensive evaluation of the system developed. As part of the optimization of the model, it was decided to use preprocessing of video data, which includes analysis of brightness and dynamic correction of brightness and contrast.

The processing times achieved by our system confirm its ability to operate in real-time. Despite using the standard YOLOv4-tiny architecture, our optimizations and lightweight preprocessing steps ensure that the model meets and exceeds the demands for real-time performance in practical applications.

In conclusion, it can be stated that the developed image object recognition system, based on the YOLO architecture and optimized according to the selected parameters, demonstrated high efficiency and accuracy. Using deep learning to define objects in images is a powerful tool that can find wide applications in various fields, from computer vision to autonomous systems. The results open up future research and development prospects in this direction. For example, in the future, the plan will use the proposed models for the dataset in [32]. Furthermore, we could explore integrating the model with other sensors, like LIDAR or radar, to improve detection accuracy in various environmental conditions, such as fog, rain, or night-time driving [35]. Additionally, we could develop adaptive algorithms to adjust the model's parameters in real-time based on environmental feedback, thus enhancing its performance in dynamic conditions.

Author Contributions: Conceptualization, N.S. and V.Y.; methodology, V.Y.; software, M.M.; validation, S.-A.M., Y.S. and S.A.; formal analysis, N.S.; investigation, M.M.; resources, M.M.; data curation, V.Y.; writing—original draft preparation, N.S. and Y.S.; writing—review and editing, S.A.; visualization, S.-A.M.; supervision, N.S.; project administration, V.Y.; funding acquisition, Y.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Research Foundation of Ukraine, project #2021.01/0103, British Academy Fellowship RaR\100727 Horizon Europe project ZEBAI: Innovative methodologies for the design of Zero-Emission and cost-effective Buildings enhanced by Artificial Intelligence (Grant Agreement ID: 101138678).

Data Availability Statement: Datasets are available by link https://www.kaggle.com/datasets/dataclusterlabs/potholes-or-cracks-on-road-image-dataset, accessed on 10 October 2022. Mysak M., Yakovyna V., Shakhovska N. (2023). Pothiles and cracks on road video and image detection system (Version 1.1.1) [Computer software]. Software Heritage, https://github.com/MysakMaksym/pothole-detection.git, accessed on 14 June 2024.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Argyroudis, S.A.; Mitoulis, S.A.; Chatzi, E.; Baker, J.W.; Brilakis, I.; Gkoumas, K.; Linkov, I. Digital technologies can enhance climate resilience of critical infrastructure. *Clim. Risk Manag.* **2022**, *35*, 100387. [CrossRef]
2. Ye, Z.; Lovell, L.; Faramarzi, A.; Ninic, J. SAM-based instance segmentation models for the automation of masonry crack detection. *arXiv* **2024**, arXiv:2401.15266.
3. Kulambayev, B.; Beissenova, G.; Katayev, N.; Abduraimova, B.; Zhaidakbayeva, L.; Sarbassova, A.; Shyrakbayev, A. A Deep Learning-Based Approach for Road Surface Damage Detection. *Comput. Mater. Contin.* **2022**, *73*, 3403–3418. [CrossRef]
4. Bučko, B.; Lieskovská, E.; Zábovská, K.; Zábovský, M. Computer vision based pothole detection under challenging conditions. *Sensors* **2022**, *22*, 8878. [CrossRef]
5. Hacıefendioğlu, K.; Başağa, H.B. Concrete road crack detection using deep learning-based faster R-CNN method. *Iran. J. Sci. Technol. Trans. Civ. Eng.* **2022**, *46*, 1621–1633. [CrossRef]
6. Long, X.Y.; Zhao, S.K.; Jiang, C.; Li, W.P.; Liu, C.H. Deep learning-based planar crack damage evaluation using convolutional neural networks. *Eng. Fract. Mech.* **2021**, *246*, 107604. [CrossRef]
7. Gagliardi, A.; Staderini, V.; Saponara, S. An Embedded System for Acoustic Data Processing and AI-Based Real-Time Classification for Road Surface Analysis. *IEEE Access* **2022**, *10*, 63073–63084. [CrossRef]
8. Heidari, M.J.; Najafi, A.; Borges, J.G. Forest roads damage detection based on deep learning algorithms. *Scand. J. For. Res.* **2022**, *37*, 366–375. [CrossRef]
9. Gunasekara, S.; Gunarathna, D.; Dissanayake, M.; Aramith, S.; Muhammad, W. Deep Learning Based Autonomous Real-Time Traffic Sign Recognition System for Advanced Driver Assistance. *Int. J. Image Graph. Signal Process.* **2022**, *14*, 70–83. [CrossRef]
10. Sami, A.A.; Sakib, S.; Deb, K.; Sarker, I.H. Improved YOLOv5-Based Real-Time Road Pavement Damage Detection in Road Infrastructure Management. *Algorithms* **2023**, *16*, 452. [CrossRef]
11. Chen, F.C.; Jahanshahi, M.R. NB-CNN: Deep learning-based crack detection using convolutional neural network and Naïve Bayes data fusion. *IEEE Trans. Ind. Electron.* **2017**, *65*, 4392–4400. [CrossRef]
12. Khemlapure, V.; Patil, A.; Chavan, N.; Mali, N. Product Defect Detection Using Deep Learning. *Int. J. Intell. Syst. Appl.* **2024**, *16*, 39–54. [CrossRef]
13. Ali, L.; Alnajjar, F.; Jassmi, H.A.; Gocho, M.; Khan, W.; Serhani, M.A. Performance evaluation of deep CNN-based crack detection and localization techniques for concrete structures. *Sensors* **2021**, *21*, 1688. [CrossRef]

14. Arya, D.; Maeda, H.; Ghosh, S.K.; Toshniwal, D.; Sekimoto, Y. RDD2020: An annotated image dataset for automatic road damage detection using deep learning. *Data Brief* **2021**, *36*, 107133. [CrossRef]
15. Bayati, M.; Çakmak, M. Real-Time Vehicle Detection for Surveillance of River Dredging Areas Using Convolutional Neural Networks. *Int. J. Image Graph. Signal Process.* **2023**, *15*, 17–28. [CrossRef]
16. Potholes or Cracks on Road Image Dataset. Available online: https://www.kaggle.com/datasets/dataclusterlabs/potholes-or-cracks-on-road-image-dataset (accessed on 10 October 2022).
17. Maeda, H.; Sekimoto, Y.; Seto, T.; Kashiyama, T.; Omata, H. Road damage detection and classification using deep neural networks with smartphone images. *Comput. Aided Civ. Infrastruct. Eng.* **2018**, *33*, 1127–1141. [CrossRef]
18. Doshi, K.; Yilmaz, Y. Road damage detection using deep ensemble learning. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020; pp. 5540–5544.
19. Maeda, H.; Sekimoto, Y.; Seto, T.; Kashiyama, T.; Omata, H. Road damage detection using deep neural networks with images captured through a smartphone. *arXiv* **2018**, arXiv:1801.09454.
20. Jeong, D. Road damage detection using YOLO with smartphone images. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 5559–5562.
21. Khan MA, A.; Alsawwaf, M.; Arab, B.; AlHashim, M.; Almashharawi, F.; Hakami, O.; Farooqui, M. Road damages detection and classification using deep learning and UAVs. In Proceedings of the 2022 2nd Asian Conference on Innovation in Technology (ASIANCON), Ravet, India, 26–28 August 2022; pp. 1–6.
22. Fedushko, S.; Shumyliak, L.; Cibák, L.; Sierova, M.O. Image Processing Application Development: A New Approach and Its Economic Profitability. In *Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies*; Springer: Berlin/Heidelberg, Germany, 2024; Volume 208, pp. 165–189. [CrossRef]
23. Kabir, M.; Kabir, A.; Rony, J.; Uddin, J. Drone Detection from Video Streams Using Image Processing Techniques and YOLOv7. *Int. J. Image Graph. Signal Process.* **2024**, *16*, 83–95. [CrossRef]
24. Adarsh, P.; Rathi, P.; Kumar, M. YOLO v3-Tiny: Object Detection and Recognition using one stage improved model. In Proceedings of the 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 6–7 March 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 687–694.
25. Hamdi, D.; Elouedi, I.; Nguyen, M.; Hamouda, A. A Conic Radon-based Convolutional Neural Network for Image Recognition. *Int. J. Intell. Syst. Appl.* **2023**, *15*, 1–12. [CrossRef]
26. Du, Y.; Pan, N.; Xu, Z.; Deng, F.; Shen, Y.; Kang, H. Pavement distress detection and classification based on YOLO network. *Int. J. Pavement Eng.* **2021**, *22*, 1659–1672. [CrossRef]
27. Henderson, P.; Ferrari, V. End-to-end training of object class detectors for mean average precision. In Proceedings of the Computer Vision–ACCV 2016: 13th Asian Conference on Computer Vision, Taipei, Taiwan, 20–24 November 2016; Part V 13. Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 198–213.
28. Khoje, S.; Bodhe, S. Comparative performance evaluation of size metrics and classifiers in computer vision based automatic mango grading. *Int. J. Comput. Appl.* **2013**, *61*, 1–7. [CrossRef]
29. Road Damage Detection Based on Deep Learning. Available online: https://www.researchgate.net/publication/372769154_Road_Damage_Detection_Based_on_Deep_Learning (accessed on 16 June 2024).
30. Spencer, B.F., Jr.; Hoskere, V.; Narazaki, Y. Advances in computer vision-based civil infrastructure inspection and monitoring. *Engineering* **2019**, *5*, 199–222. [CrossRef]
31. Huang, Z.; Chen, W.; Al-Tabbaa, A.; Brilakis, I. NHA12D: A new pavement crack dataset and a comparison study of crack detection algorithms. *arXiv* **2022**, arXiv:2205.01198.
32. Žeger, I.; Grgic, S.; Vuković, J.; Šišul, G. Grayscale image colorization methods: Overview and evaluation. *IEEE Access* **2021**, *9*, 113326–113346. [CrossRef]
33. Zhang, S.; Li, P.; Xu, X.; Li, L.; Chang, C.C. No-reference image blur assessment based on response function of singular values. *Symmetry* **2018**, *10*, 304. [CrossRef]
34. Ouyang, H. DEYO: DETR with YOLO for End-to-End Object Detection. *arXiv* **2024**, arXiv:2402.16370.
35. Zhang, Y.; Liu, C. Real-Time Pavement Damage Detection with Damage Shape Adaptation. *IEEE Trans. Intell. Transp. Syst.* **2024**, 1–10. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.