Applied Computer Science 2025, vol. 21, no. 1, pp. 164–181
https://doi.org/10.35784/acs_6692

Submitted: 2024-10-11 | Revised: 2025-01-03 | Accepted: 2025-01-06

Keywords: artificial neural network, limited resources inference, road defect detection, embedded device

Paweł TOMIŁO [1*]
[1] Lublin University of Technology, Poland, p.tomilo@pollub.pl
[*] Corresponding author: p.tomilo@pollub.pl

LANA-YOLO: Road defect detection algorithm optimized for embedded solutions

Abstract

Poor pavement condition leads to increased risk of accidents, vehicle damage, and reduced transportation efficiency. The author points out that traditional methods of monitoring road conditions are time-consuming and costly, so a modern approach based on the use of developed neural network model is presented. The main aim of this paper is to create a model that can infer in real time, with less computing power and maintaining or improving the metrics of the base model, YOLOv8. Based on this assumption, the architecture of the LANA-YOLOv8 (Large Kernel Attention Involution Asymptotic Feature Pyramid) is proposed. The model's architecture is tailored to operate in environments with limited resources, including single-board minicomputers. In addition, the article presents Basic Involution Block (BIB) that uses the involution layer to provide better performance at a lower cost than convolution layers. The model was compared with other architectures on a public dataset as well as on a dataset specially created for these purposes. The developed solution has lower computing power requirements, which translates into faster inference times. At the same time, the developed model achieved better results in validation tests against the base model.

1. INTRODUCTION

Many factors influence road accident rates, and to simplify, the entire system can be considered as a complex set of three key elements: man - machine - environment. The element of the environment itself consists of many factors, which include, among others, weather conditions, the condition of road infrastructure and even traffic density. The road surface is an important component of this environment, which has a significant impact on road safety and the number of accidents. The quality of a road surface, which includes its smoothness, adhesion, durability and weather resistance, has a significant impact on how drivers control vehicles. Although the condition of the road surface is only one of many factors influencing the number of road accidents but it is one of the key ones. Ensuring the appropriate quality of the road surface, regular maintenance and monitoring its condition are important activities aimed at improving road safety and reducing the number of accidents (Bhattacharya et al., 2022; Jakobsen et al., 2023).

Due to material deterioration brought on mainly by excessive traffic, severe weather, age, subpar construction, and improper maintenance, roadway infrastructure is prone to structural degradation. Timely diagnosis of defects and subsequent repair are crucial for the comfort and riding quality of travelers, as these factors are intrinsic to the effective operation of a road transport system (Ranyal et al., 2022). The methods for detecting road damage have evolved significantly over time, moving from simple manual inspections conducted by workers on-site to more sophisticated approaches that use automated systems and advanced image processing technologies. These modern techniques not only improve accuracy but also enhance efficiency by analyzing road conditions faster and more comprehensively (Arya et al., 2020).

The issue of creating a model for detecting road defects has been and continues to be the subject of much research. One of the most prominent examples is Global Road Damage Detection Challenge-RDDC2020 (Arya et al., 2020) and Crowdsensing-based Road Damage Detection Challenge-CRDDC'2022 (Arya et al., 2022). In both challenges the top scores were achieved by models from the YOLO family. In the 2022 challenge, the first place went to the ShiYu SeaView team with an average score of 0.716. The proposed solution was based on multi-model ensemble. The solution is based on using YOLOv5 and YOLOv7 models to achieve dense predictions and CascadeRCN with Swin Transformer as backbone to increase efficiency for small objects.

The architecture of the YOLO family of models is regularly modified by researchers to improve their performance and accuracy. In a study (Jiang, 2024), significant improvements were made using the Bottleneck block with deformable attention (Zhu et al., 2020) and slim-neck (H. Li et al., 2024) mechanisms. These innovations were aimed at increasing the precision of the model while maintaining its compact form. These techniques resulted in a 3.9 percentage point improvement in accuracy compared to the base YOLOv8 nano model. However, these modifications involved a slight compromise - the number of frames per second (FPS) processed by the model decreased by 12.14%. Nonetheless, the results show significant benefits from the modifications, especially in terms of increasing detection efficiency while maintaining the relative speed of the model. The EMG-YOLO architecture proposed by (Xing et al., 2024) allowed an increase in the metric, in the form of Mean Average Precision (mAP) for Intersection Over Union value 0.5, relative to the yolov5 base model by 2.6 pp with a significant increase in theoretical computing power requirements expressed in giga FLoating point-Operation Per second (FLOPs) by 85%. The model's architecture uses Global Context Block and a hybrid channel strategy for the detection head, as well as a modified loss function. Researchers (J. Wang et al., 2024) proposed an improved yolov8 model in small size. This model uses Lightweight bottleneck based on Efficient Multi-scale Attention (Ouyang et al., 2023) and Partial Convolution (G. Liu et al., 2018). Changes have also been made in the form of SimSPPF and Dyhead. The introduced medications allowed an increase in mAP:0.5 by 2.6 pp with a 21.67% decrease in theoretical computing power requirements relative to the YOLOv8 base model in the small size. The introduction of modifications was also associated with a 64.46% decrease in FPS.

All the solutions presented involve a trade-off between model accuracy and inference time. Inference time is not significant when the model is used in an environment with large computing resources, but for environments with limited computing resources such as embedded systems or single-board computers, inference time is significant, particularly in real-time applications. For this reason, the author decided to develop a model an artificial neural network model architecture for the task of detecting defects in road pavement - LANA-YOLO. The main goal of creating this model is to preserve or improve metrics relative to the base model while reducing FLOPs and the time required for inference on devices with limited computing power in real-time. Based on the research conducted by (Tang et al., 2023), it was decided to use YOLOv8 due to its high accuracy against the detection task. Due to the use of a single-board computer, a nano version model was adapted for testing.

The following sections focus on describing the architecture of the base model (1.1. Architecture of YOLOv8) and presenting improvements to its architecture in the context of reducing theoretical computing power requirements and increasing inference speed while maintaining or improving metrics relative to the base model - section 1.2. Overview of solutions used in machine learning models for the detection task. The rest of the article presents the architecture selection process in the context of road defect detection. Tests were carried out on part of the CRDDC'2022 set, and then validation was performed on the new data. Based on the tests, a new Poland Road Condition - Detection Dataset (PRCDD) was created, on which the developed model was also tested.

1.1. Architecture of YOLOv8

The YOLOv8 architecture consists of 3 basic elements: Backbone, Neck, and Head. Features are extracted and encoded from the input data by Backbone. The YOLOv8 architecture backbone uses blocks that benefit from a faster implementation of Cross Stage Partial Bottleneck with 2 convolution - C2F. The shortcut designation shown in Figure 1 corresponds to the use of the residual connection in the C2F block bottlenecks. The use of this block allows for increased generalization capabilities of the model (Bai et al., 2023). Neck objective is to provide more informative feature representations and enhance the extracted features from the backbone. The YOLOv8 architecture uses a faster implementation of Spatial Pyramid Pooling - SPPF. This layer, in simple terms, feeds the output tensor from the backbone through the following max pooling layer, and then combines the output tensors from each layer. Comprising task-specific layers, the head generates the final prediction or inference by utilizing the data retrieved by the Neck and Backbone. In the case of yolov8, a decoupled head was used (Xu, 2022). The architecture diagram of YOLOv8 is shown in Figure 1.

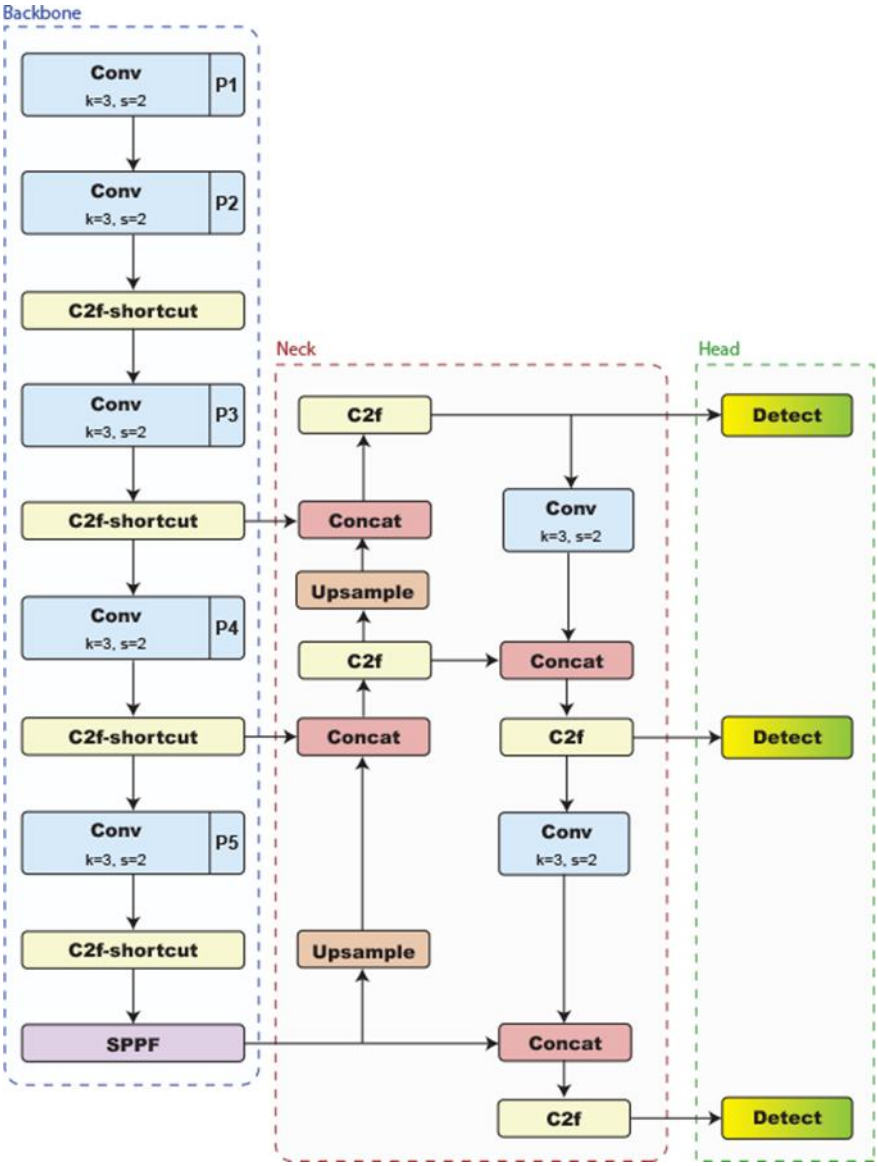

Fig. 1. Simplified architecture diagram of YOLOv8

1.2. Overview of solutions used in machine learning models for the detection task

In the presented section, the emphasis is on the module ensuring increased prediction quality with a reduced number of model parameters. For this reason, the following layers and solutions were selected:

- Large Scale Kernel Attention;
- Involution;
- Asymptotic Feature Pyramid Network (neck).
- Ghost convolution (backbone).

The Large Scale Kernel Attention (LSK-A) module is based on the decomposition of a larger kernel into a depth-wise convolutions sequence, in such a way that in each subsequent decomposition the kernel size and dilatation rate increase. The use of increasing hyperparameters ensures sufficient rapid expansion of receptive field. The discussed module, compared to the use of a single convolution layer with a larger kernel, ensures easier kernel selection in later stages and is more efficient. The effectiveness lies in the fact that while maintaining the same receptive field, the number of model parameters is smaller. The use of a layer of this type allows for greater concentration on the most relevant features. In the module, the information from the input feature map X is processed by successive decomposed depth-wise convolutions LK_i, by which tenosors $\widetilde{U}_i$ are obtained, which are further concatenated with each other to form the tensor $\widetilde{U}$. In the Spatial kernel selection (SK) block, the $\widetilde{U}$ tensor is first subjected to channel-based average pooling $F_{avg}(\cdot)$ and channel-

based maximum pooling $F_{max}(\cdot)$ operations separately. Next, the data is concatenated using convolution $C^{2\to N}$ to create N spatial attention maps, allowing information sharing between different spatial descriptors (Y. Li et al., 2023). The entire discussed process is described by equation (1):

$$SK = C^{2\to N}\left[F_{avg}(\widetilde{U}), F_{max}(\widetilde{U})\right] \tag{1}$$

For each of the created SK in order to obtain individual spatial selection mask (SSM), sigmoidal function σ was used. Sequences of decomposed large kernels are weighted based on corresponding SSMs, and then values are combined by applying a convolution layer C. The whole process allows to obtain attention feature S - equation (2):

$$S = C\left(\sum_{i=1}^{N} \sigma(\widetilde{SK}_i) \cdot \widetilde{U}_i\right) \tag{2}$$

The output value Y from the LSK-A layer is the element-wise product of X - input tensor and S - attention features. A simplified diagram of the LSK-A layer is shown in Figure 2.

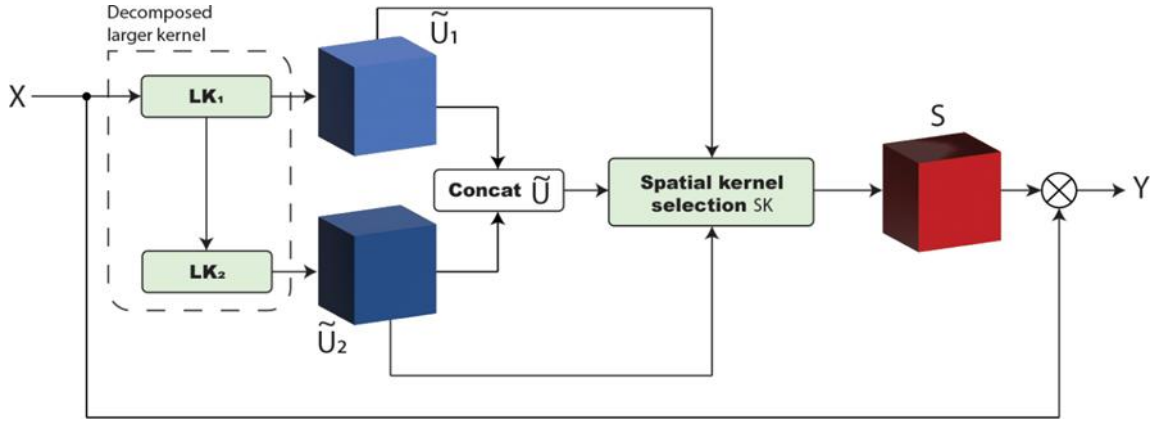

Fig. 2. Simplified diagram of the LSK-A layer

Researchers (X. Wang et al., 2023) proposed the use of the LSK-attention layer, which allowed, in their case, to increase the parameter mAP@0.5% from 87.4 to 88.7. The discussed layer with the use of Simple Spatial Pyramid Pooling Fast - SimSPPF in the previous layer allowed to reduce the number of model parameters while achieving higher prediction accuracy. SimSPPF increases the perceptual field of view by employing a cascade of several small-sized pooling kernels rather of a single large-sized pooling kernel. (J. Liu et al., 2023) One technique that can help reduce parameters and improve model accuracy is layer involution. The kernel of the involution operation differs in the spatial dimension, but is common to all channels. This means that involution has inverse characteristics to convolution operations. The involution kernel, which belongs to a certain spatial location, is generated depending on the input feature vector at the appropriate location, which makes it possible to reduce the redundancy of kernels by sharing along channel dimension. The involution operation is able to model long-range interactions and adaptively allocate weights for different positions which allows higher weights to be given to more significant features, similar to self-attention, except that the involution kernel generation is conditioned on a single pixel rather than based on the relationship with neighboring pixels (D. Li et al., 2021). The involution operation is described by Equation 3 and schematically shown in Figure 3a.

$$Y_{i,j,k} = \sum_{(u,v)\in\Delta_K} \varkappa_{i,j,u+\left[\frac{K}{2}\right],v+\left[\frac{K}{2}\right],\left[\frac{kG}{C}\right]} X_{i+u,j+v,k} \tag{3}$$

where: G – numbers of groups;
$X \in \mathbb{R}^{H\times W\times C}$ – input tensor;
Δ_K – set of offsets in the vicinity taking into account the central pixel's convolution;
K – kernel size;
$\varkappa_{i,j}$ – kernel generation function (equation 4).

$$\varkappa_{i,j} = \phi(X_{ij}) = W_1\sigma(W_0 X_{ij}) \tag{4}$$

where: $\phi: \mathbb{R}^C \rightarrow \mathbb{R}^{K\times K\times G}$ – kernel generation function;
$X_{i,j}$ – single pixel;
σ - Batch Normalization and in my case Sigmoid Linear Unit (SiLu) function.

Asymptotic Feature Pyramid Network (AFPN) is an architecture used in object detection for detecting features at different scales. The AFPN architecture features asymptotically integrates features at different scales. The AFPN uses the Adaptively Spatial Feature Fusion (ASFF) layer. In the discussed layer, the first operation is feature resizing. This operation relies on features being up-sampled or down-sampled to match expected tensor dimensions. In the case of up-sampling functions, bilinear interpolation is used, and in the case of down-sampling functions, the features are fed, for example, with a 2 × 2 convultion with a step of 2 to get a 2-fold down-sampling, and so on. The next step is Adaptive Fusion (AD), the operation involves applying spatial importance weight $W_{x_{ij}^{n\rightarrow l}}$ to each output feature vector $X_{ij}^{n\rightarrow l}$, where (i,j) denotes feature vector position and $n \rightarrow l$ denotes feature resizing from level n to level l (S. Liu et al., 2019; Yang et al., 2023). AD operations are represented by equation (5):

$$y_{ij}^{l} = \sum W_{X_{ij}^{n\rightarrow l}} \cdot X_{ij}^{n\rightarrow l} \tag{5}$$

The weights are given in such a way that equation (6) must be satisfied:

$$\sum W_{X_{ij}^{n\rightarrow l}} = 1 \tag{6}$$

A schematic representation of the operations discussed in AFPN is shown in Figure 3b.

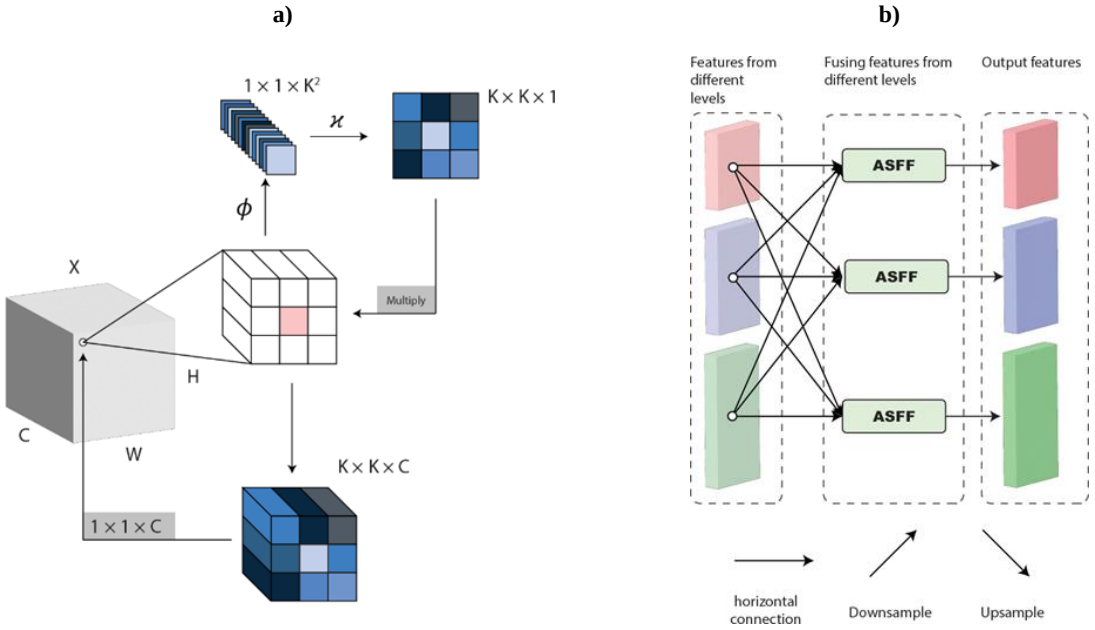

Fig. 3. a) Involution, b) AFPN

In classical convolution, a set of filters $f \in \mathbb{R}^{c\times k\times k\times n}$ (where: c - number of channels, $k \times k$ - kernel size, n - number of filters) is applied to the input feature map $X \in \mathbb{R}^{c\times h\times w}$ (where: h - height, w – width) according to equation (7).

$$Y = X * f \tag{7}$$

where: $Y \in \mathbb{R}^{c\times h'\times w'}$ - output feature map;

$*$ - convolution operation;
h' - height of output tensor;
w' - width of output tensor.

The property of the output Convolutional layer maps frequently have a lot of redundancy, and some of them may be comparable to one another. In addition, the Floating Operations Per second (FLOPs) of a layer of this type is usually large due to the number of filters and channels. The number of FLOPs for a convolutional layer can be calculated by applying equation (8) (Han et al., 2020).

$$O(\cdot_C) = n \cdot h' \cdot w' \cdot c \cdot k \cdot k \tag{8}$$

where:

$\cdot_C$ – list of parameters of a given convolution layer C.

According to (Han et al., 2020) generating these redundant feature maps one by one with large number of FLOPs and parameters is unnecessary. Ghost Convolution can be used instead. The idea behind this solution is to use fewer filters and, based on them, generate so-called Ghost features using simple linear transformations. In this method, a feature map is first generated according to equation (9).

$$Y' = X * f' \tag{9}$$

where: $Y' \in \mathbb{R}^{c \times h' \times w'}$ - intrinsic feature maps
$f' \in \mathbb{R}^{c \times k \times k \times m}$ - filters;
$m \geq n$ – number of filters;

Then, in order to obtain the target number of map features n, a series of operations requiring low computing power resources should be applied, according to equation (10).

$$y_{ij} = \Phi_{i,j}(y_i'), \forall\, i = 1, \ldots, m, j = 1, \ldots, s \tag{10}$$

where: s – number of ghost features;
y_i' denotes i-th intrinsic feature map in Y';
$\Phi_{i,j}$ – operations requiring low computing power resources (e.g. linear transformation) for generating j-th ghost feature maps y_{ij} (except last one).

The $\Phi_{i,s}$ operation is responsible for keeping the previously created y_i' map unchanged (identity mapping). The principle of the Ghost Convolution block is shown graphically in Figure 4.

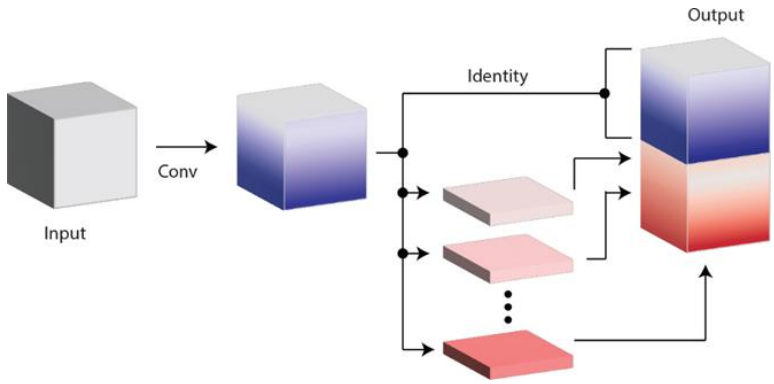

Fig. 4. Schematic of the operation of the Ghost Convolution block

1.3. Contribution of this study

The data set that was chosen to train the model represents one of the more difficult challenges in computer vision. As indicated in the study (Arya et al., 2022), the top ten models running on this data set from Japan achieved an average mAP50 metric of 0.7278. However, these models were not evaluated in terms of inference time or applicability to real-time systems, as computing power requirements constraints were not a priority during their development. In most cases, the leading solutions were based on a multi-model ensemble approach, which involved using one or more models simultaneously. While this approach provides high

performance in terms of detection accuracy, it also leads to a significant increase in inference time and computational resource requirements, making it impractical for real-time applications. In response to these challenges, this study proposes a novel solution based on YOLO family model architectures. The main goal of the project was to provide a trade-off between high efficiency and low computing power requirements, making the model suitable for embedded applications. The work analyzed and implemented advanced techniques such as Large Scale Kernel Attention, Involution, Asymptotic Feature Pyramid Network, SimSPPF and Ghost Convolution. What's more, an innovative Basic Involution Block was developed to enable effective feature recalibration through the involution mechanism, which improves the model's ability to generalize and detect complex structures. The developed model has been designed for seamless conversion to TensorRT format, which allows for significant acceleration of inference on embedded devices such as NVIDIA Jetson Xavier NX. The performance tests carried out confirmed the model's ability to achieve high values of metrics under conditions of limited hardware resources. Compared to the baseline model, the proposed architecture allows a significant reduction in the number of parameters while improving performance metrics. The developed model makes an important contribution to the state-of-the-art, as it combines computational efficiency, innovative optimization approaches and applicability to real embedded systems. Such a model is particularly relevant in the context of the growing demand for artificial intelligence technologies capable of operating in real time on devices with limited resources.

In addition, a data set described in Section 3.2 Model training and validation - CoCGRCDD - was developed specifically for the purpose of adapting the model and model evaluation to road defects occurring in Poland. It should be noted that there is no publicly available data set of road defects from Poland in the literature. The developed solution was optimized and fine-tuned for the detection of road defects in pavements occurring in Poland.

2. METHOD

The main objective of the research assumed in the publication is to create an artificial neural network (ANN) model, which will be able to infer as quickly as possible on devices with limited resources. This device is understood as a single-board minicomputer, where model inference will be performed from the Central Processing Unit (CPU) or a specially adapted Graphics Processing Unit with multi-threaded processors. Based on the literature review, YOLOv8 was selected as the base architecture of the ANN model. Due to limited computational resources, the “n” architecture variant was selected, which has the lowest computational power requirements (Q. ; Liu et al., 2023).

In order to optimize the discussed model, based on the previous achievements of researchers, it was decided to test the following main solutions:

- use of Ghost blocks in the backbone
- use of neck in the form of AFPN, AFPN with GhostV2 and AFPN with BIB block.
- checking the change of SPPF to SimSPPF and adding layers of LSK, LSK-A and involution blocks in different places of the architecture.

The RDD2022 dataset (Arya et al., 2022) and more precisely sub-dataset of photos from Japan where used to train the developed neural network model architectures. It was decided to choose this part of the data set because of the similar conditions to those on which the model will be tested. From the discussed data set, the classes D00 - Longitudinal Crack, D10 - Lateral Crack, D20 - Alligator Crack, D40 - Pothole were used to train the model.

The data set was divided in a base ratio of 7:3, taking into account the balanced occurrence of instances of the class in both sets. Therefore, the real share of images from the entire data set in the training set, and the validation set, respectively, is 68.66% and 31.34%. The share of each class in both sets is shown in Figure 5.

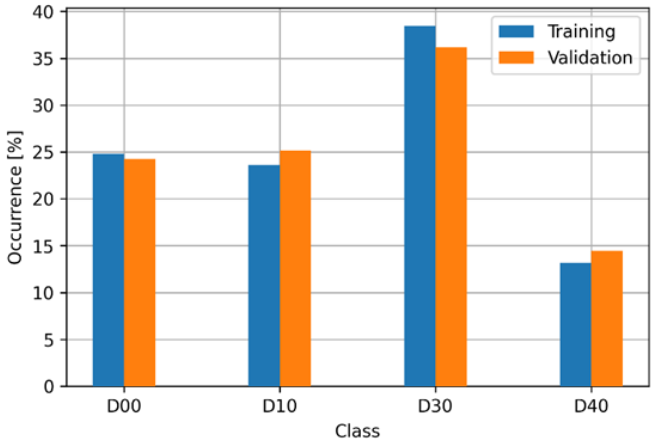

Fig. 5. Division of the data set and the number of instances of a given class

2.1. Architecture of the LANA-YOLO model

The developed LANA-YOLO architecture used a slightly modified YOLOv8 backbone. The SPPF block was replaced by SimSPPF, which translated into a reduction of model parameters and allowed the use of, following, the LSKA block. The use of an additional block, which by default is not present in the architecture, affected the increase in model parameters, but through the subsequent element reducing the number of parameters and theoretical computational complexity, it was possible to achieve a smaller number of parameters than in the case of the base model. The main change in the architecture is the use of neck in the form of AFPN. A Basic Involution Block (BIB) was developed for the model, which was applied after the ASSF blocks in the AFPN. The developed BIB uses the involution layer and the following layers of batch normalization and SiLu activation function, in the next stages the convolution layer and batch normalization were applied, the input tensor is passed through the previously presented layers, and after the transformation the input tensor is added to the output tensor, the last stage is the SiLu activation function. Equation 9 shows the described process of the BIB block:

$$Y_{BIB} = \sigma\left(X + B_n\left(C\left(\phi(X)\right)\right)\right) \tag{11}$$

where: $X \in \mathbb{R}^{H\times W\times C}$ – input tensor;
ϕ – previously described involution function;
C – convolution function;
B_n – batch normalization;
σ – SiLU activation function.

The main goal of the proposed architecture is to achieve high accuracy in defect detection while minimizing the demand for computational resources. To this aim, the architecture uses advanced components, each of which has been selected to improve efficiency in specific aspects of the detection process. The model uses the LSK-A layer, which significantly enhances the network's ability to capture global road pavement features. Traditional convolutional layers operating on small kernel sizes can have difficulty effectively detecting extensive or irregular damage, such as cracks or cavities covering large areas. By using a larger kernel size, the LSK-A layer increases the field of view of the convolution operation to better capture these features. In addition, the built-in attention mechanism allows selective highlighting of the most relevant spatial features, improving the overall reliability and accuracy of the model. An AFPN was used as the Neck of the model, which facilitates the fusion and optimization of features from different levels of the network. Pavement damage can come in different shapes and sizes, requiring efficient fusion of feature representations from different layers. AFPN achieves this by adaptively fusing features from different levels of the network hierarchy, enhancing the model's ability to detect both small and subtle damage and large and distinct damage. At the same time, AFPN's optimized design reduces the computational load, which supports the goal of minimizing computing power requirements. The SimSPPF module supports the efficient creation of feature pyramids, which is crucial for damage detection at different scales. By aggregating feature maps from different layers,

SimSPPF makes it possible to capture spatial hierarchy, which is important for distinguishing damage types and their characteristics. Its simplified architecture reduces computational complexity compared to traditional feature pyramid methods. An innovative solution in the model under study is the use of BIB. The involution layer is particularly effective in analyzing complex textures and patterns specific to road pavement. Integrating the BIB into the model improves the network's ability to learn complex spatial relationships while maintaining computational efficiency. The adaptive nature of the involution layer enhances the flexibility and versatility of the model, which is crucial for detecting a wide range of defects under varying environmental conditions. A diagram of the described structure of the LANA-YOLO model is shown in Figure 6.

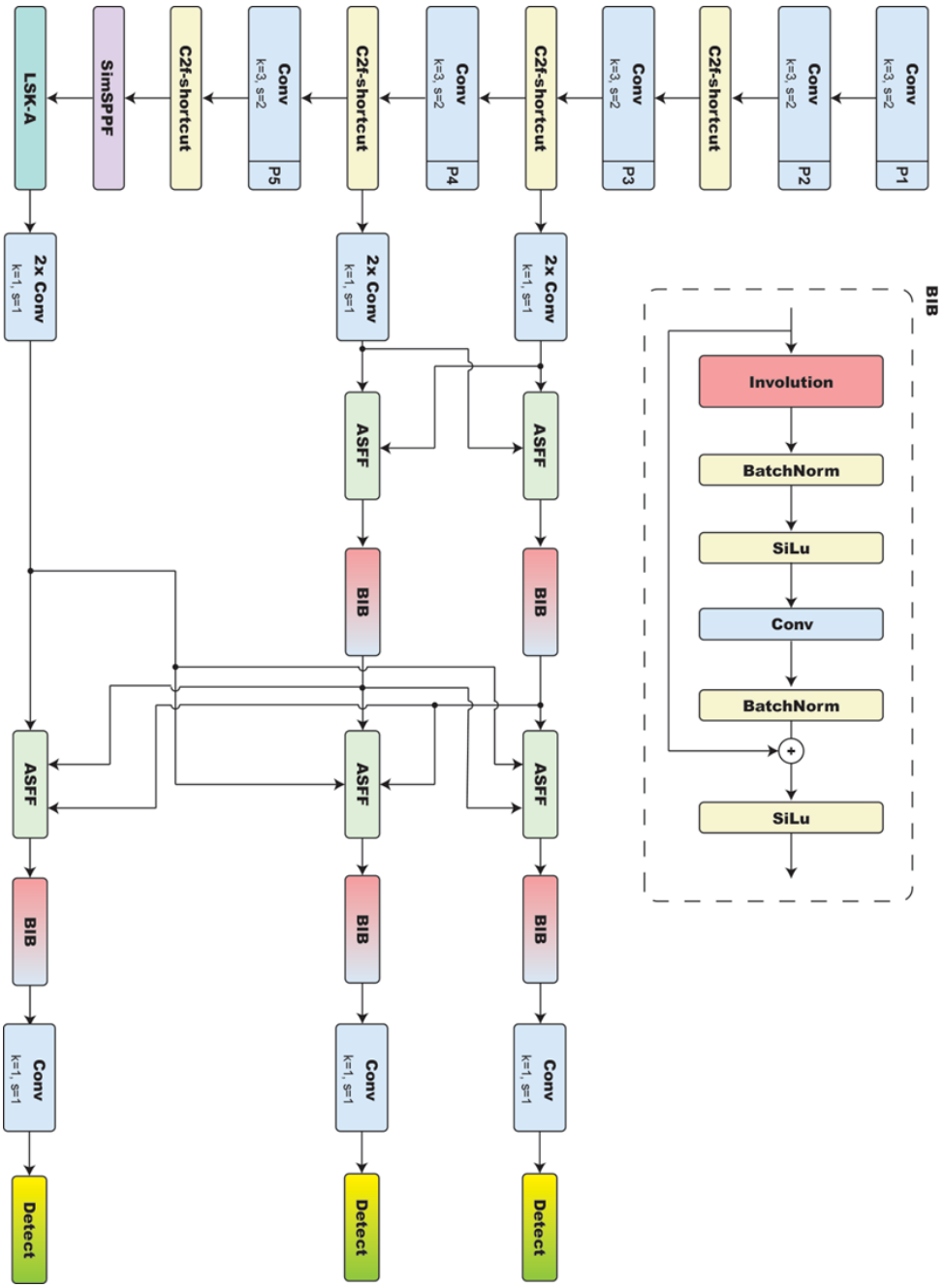

Fig. 6. Architecture of LANA-YOLO

2.2. Training results and model description

The developed neural network model architectures were trained on the previously described subset of CRC2022 dataset. The learning process was carried out hybrid, so for 280 epochs the model was learned using mosaic augmentation, and for another 20 images were fed to the model with augmentation in the form of

changing the value of HSV - H 0.015, S: 0.7, V: 0.4.This procedure was used to stabilize the gradient. Mosaic augmentation based on a set of 4 images creates a mosaic, where images are merged together, as shown in Figure 7.

Fig. 7. Example of mosaic augmentation

In addition to the base model, 8 models were tested, for which different architectures were developed. Two of the developed models used ghost convolution layers in their backbone. The use of these layers made it possible to firmly reduce the parameters as well as the theoretical computing power requirements expressed in GFLOPs (Giga Floating Point Operations Per Second). In the first "involution (last)" model, involution layers were applied in the neck before the head itself. The next model "GhostAFPN + involution (last)" used a neck in the form of AFPN with ghost convolution layers between ASSFs. The model in question had the smallest number of parameters as well as GFLOPS, which were 1625628 and 5.0, respectively. However, this was associated with a decrease in mAP50 accuracy relative to the base model. Another models were based on the unmodified backbone YOLOv8. Models using AFPN, with a change from SPPF to SimSPPF using Large Scale Kernel Convolution and Large Scale Kernel Attention were tested. The application of the AFPN layer to the YOLOv8 model alone reduced the parameters and increased the accuracy of mAP50. The highest accuracy of both mAP50 and mAP50-95 was distinguished by the developed LANA-YOLO model, which uses LSKA and AFPN with the developed Basic Involution Block (the exact architecture of the model is described in Section 5). The exact parameters of the developed models are shown in Table 1.

Tab. 1. Comparison of the developed models

Backbone	**Model**	**Parameters**	**GFLOPs**	**mAP 50**	**mAP 50-95**
Yolov8	Baseline (YOLOv8n)	3006428	8.9	0.536	0.270
Ghost	involution (last)	1943058	5.5	0.509	0.248
	GhostAFPN+ involution (last)	1625628	5.0	0.509	0.258
Yolov8	AFPN	2290537	6.7	0.539	0.267
	(SimSPPF + LSK) + AFPN	2410854	6.8	0.536	0.264
	(SimSPPF + LSKA) + AFPN	2540975	6.9	0.534	0.264
	(SimSPPF + involution) + AFPN	2566889	7.0	0.517	0.260
	SimSPPF + LSKA + AFPN + involution (last)	2542438	6.9	0.536	0.264
	LANA-YOLO	2490617	6.8	0.542	0.278

* The parameters of all models presented in the table above (Table 1) were calculated without combining the Conv2d and BatchNorm2d layers

According to the results shown in Table 1, the use of AFPN alone has a positive effect on reducing computing power requirements while increasing the mAP50 metric relative to the base model. However, this is associated with a decrease in the mAP50-95 metric. Changing the backbone to one that incorporates Ghost convolution layers firmly allows for a reduction in theoretic computing power requirements, however, this translates into a rather large decrease in mAP50. The further use of a neck in the form of AFPN with ghost convolution allowed an even greater reduction in theoretical computing power requirements while maintaining the same values of the mAP50 metric relative to the model using a backbone with ghost convolution and layers of involution before detection head. In addition, it allowed for an increase in the mAP50-95 metric.

The model with the highest mAP was selected and then compared more accurately to the base model by applying confusion matrix and F1-score metrics. The base YOLOv8 model correctly identified 0.42 cases from class D00, 0.44 from class D10, 0.62 from class D20 and 0.46 from class D40, with an F1-score of 0.52. In contrast, the proposed model achieved an F1-score of 0.53 and correctly identified 0.45 cases from class D00, 0.47 from class D10, 0.64 from class D20 and 0.5 from class D40. The values in question are shown in Figure 8 as a confusion matrix for both models and Table 2 shows the F1-score for both models by class.

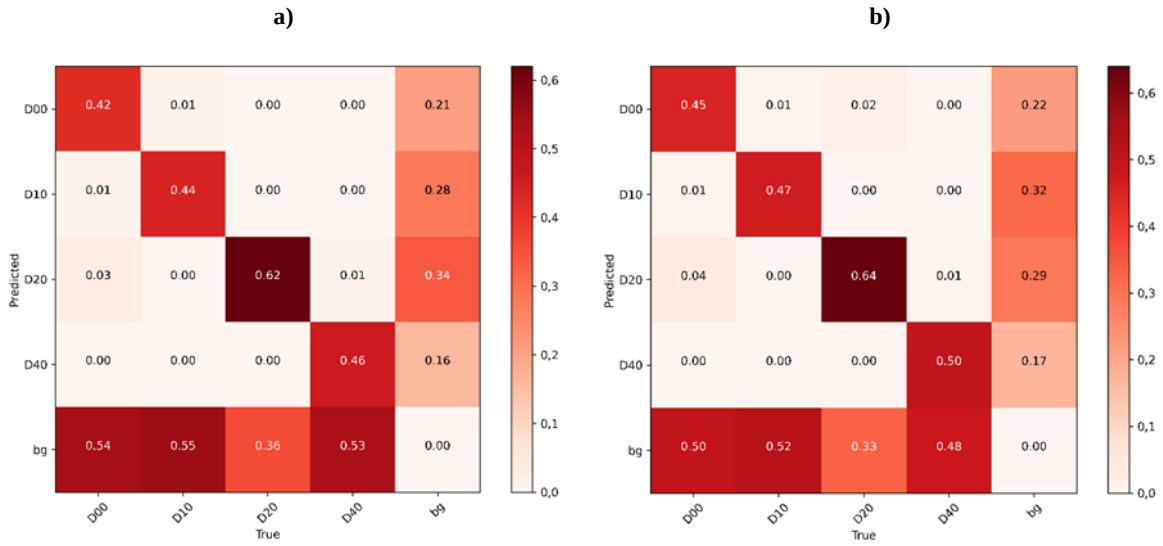

Fig. 8. Comparison of confusion matrices: a) YOLOv8n; b) LANA-YOLOv8

Tab. 2. Comparison of the developed models

Class	F1-score (YOLOv8)	F1-score (LANA-YOLO)
D00	0.47	0.49
D10	0.48	0.48
D20	0.61	0.63
D40	0.50	0.51

In order to illustrate the quality of the predictions of both models, Figure 9 shows a comparison of the predictions against the true values. The prediction accuracy level was set at $C \geq 0.3$. From the example presented below, the YOLOv8n model was unable to determine any road defect for the first and last samples. In the case of the second sample, the model correctly determined several bounding boxes, but its precision and prediction accuracy were lower than in the case of the developed LANA-YOLO model.

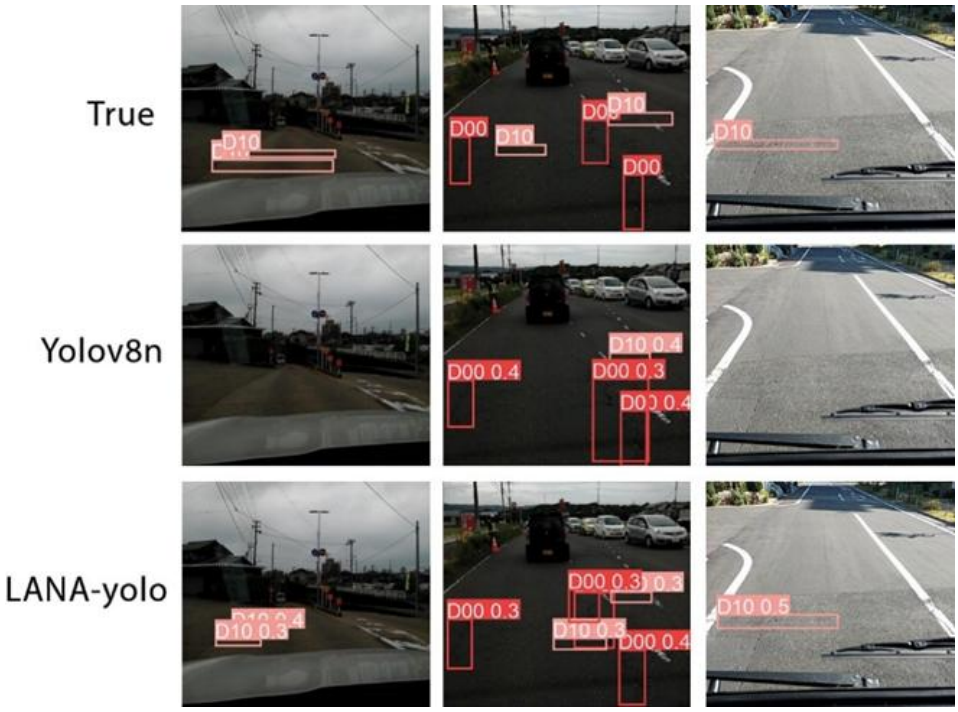

Fig. 9. Example predictions of the base model and the developed LANA-yolo model for the validation set

3. RESULTS AND DISCUSSION

3.1. Models conversions and tests

In order to accelerate the ability to infer the model on devices with limited resources, the created models were converted to the ONNX format for inference on the CPU, and in the case of inference on the GPU, the TensorRT (Zhou & Yang, 2022) format was used. The floating point accuracy of the models was reduced from float32 to float16, therefore validation was performed again to detect potential accuracy drops - no major differences in the quality of inferences were found. Tests of the converted models were carried out on an AMD Ryzen 9 900X processor - ONNX model, a single-chip Raspberry pi 5 minicomputer - ONNX model, and on a embedded device equipped with a GPU adapted for inference based on the AI model - Jetson Xavier NX - TensorRT model. The LANA-YOLO model achieved an inference speed of 43.40ms on the Ryzen processor, while the inference time on the yolov8n model was 5.5ms longer. In the case of Raspbery pi 5, the inference time difference was 2.27ms in favour of the LANA-yolo model. In the last inference test on Jetson Xavier, the difference in inference times was the largest, LANA-YOLO was 5.9ms faster. All tests performed only take into account the model inference time, without taking into account the pre-processing and post-processing time. The average inference time was calculated based on the network processing the same image 10 times with batch = 1, in order to obtain consistent results. Table 3 shows the summary of inference times for various devices.

Tab. 3. Model inference time with batch equals to 1 on different devices

Format	Model	AMD Ryzen 9 3900X	Raspberry pi 5	Jetson Xavier NX
ONNX	Yolov8n	48.90ms	305.00ms	-
	LANA-yolo	43.40ms	302.73ms	-
TRT	Yolov8n	-	-	41.15ms
	LANA-yolo	-	-	35.25ms

In addition to performance tests, power demand tests were conducted for the Jetson Xavier NX embedded device according to the procedure described in (Tomiło et al., 2024), the standard YOLOv8 architecture achieved a power demand of 4.87 W, while the developed LANA-YOLO architecture achieved a computing power demand of 4.31 W.

An experiment was carried out based on photos from a camera installed in a car. The experiment was carried out for various weather conditions and at different times of the day. It was noticed that the predictions of the base model have a lower level of confidence for the same objects compared to the LANA-YOLO model. The base model also showed a greater tendency to false positives, and at the same threshold it did not detect objects that were detected by the LANA-YOLO model. A visual comparison of the model predictions is shown in Figure 10.

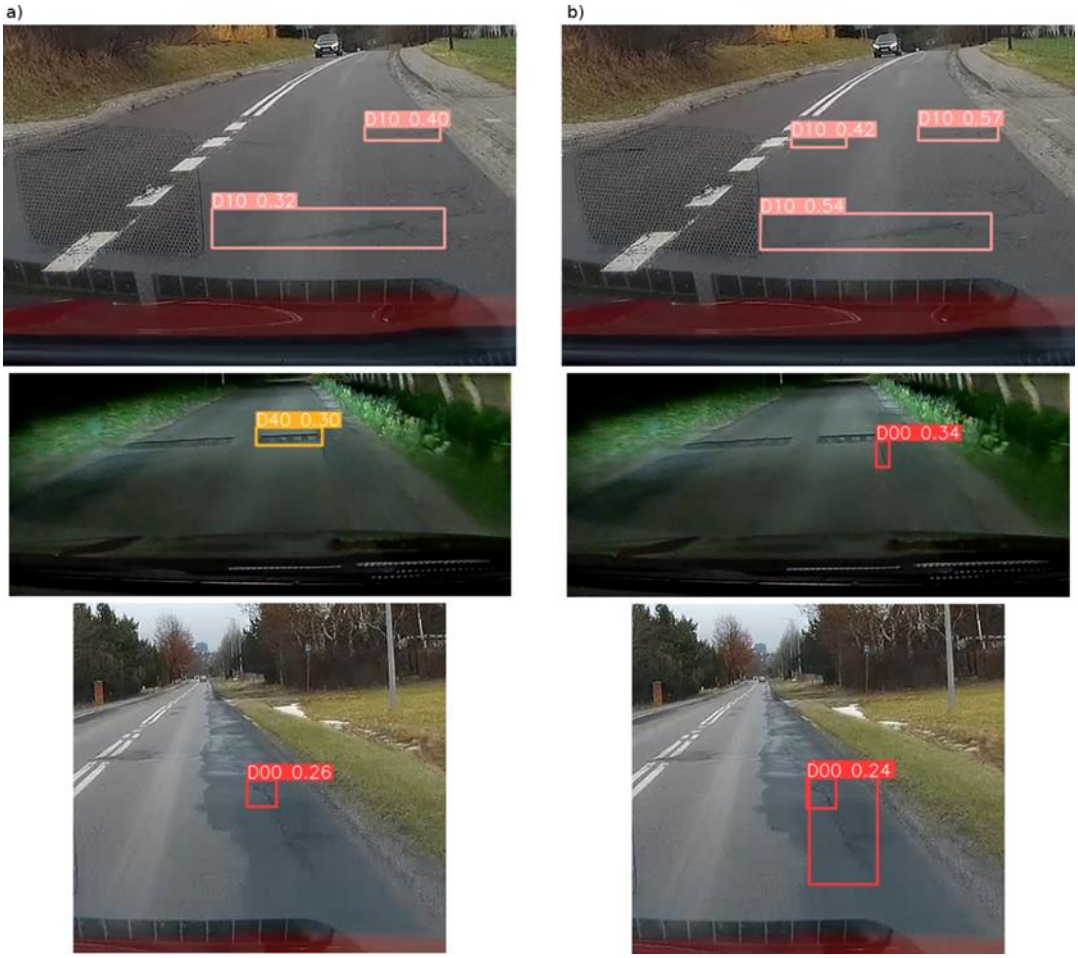

Fig. 10. Comparison of the prediction quality for the base model and the developed LANA-YOLO model

During the experiments, it was also noticed that the current image quality from the camera is insufficient, and additional elements of the environment that are in the camera frame are not necessary for inference in any way. This means that the optimal solution is to cut out the part of the image containing the road surface in order to increase the accuracy of the model. For this purpose, a camera with a higher resolution should be used, which will ensure the appropriate quality of the image section.

Additionally, tests have shown that locating the camera near the vehicle's windshield is not an optimal solution. A large part of the surroundings, apart from the road itself, is located inside the frame. Obtaining sections of satisfactory quality requires the use of a camera with a higher resolution, and in addition, it will be necessary to cut out the appropriate region of the interest, which increases the inference time.

3.2. Model training and validation – CoCGRCDD

Due to the previously presented issue, it was decided to create a new Camera on Car Grille Road Condition - Detection Dataset (CoCGRCDD) (Tomiło, 2024). The dataset was created on the basis of images from a camera placed on the grille of the car. This kind of solution allowed the use of frames from the recordings at a resolution of 1920x1080px without any manipulation other than scaling. The recordings were made using a USB camera placed on the car's grille - Figure 11. The developed dataset is based on road surface images from Poland's Lubelskie voivodeship. The location of the camera is shown in Figure 11.

Fig. 11. Location of the camera

The images were annotated using the Computer Vision Annotation Tool (CVAT) (cvat-ai / cvat, 2025).

The set implements similar classes to the CRCDD set, but they have been extended with additional elements in order to adapt to the pavement damage occurring on Polish roads, for this reason, the following classes appear in the developed data set:

- C01 - Longitudinal cracks and pronounced discontinuity of the material structure in the longitudinal axis;
- C02 - Transverse cracks and pronounced discontinuity of the material structure in the transverse axis;
- C03 - Alligator cracks and delamination of the surface layer occurring in their area;
- C04 - holes on the road surface and larger cavities erosion (such as in the area of cracks).

Examples of the classes in question are shown in Figure 12.

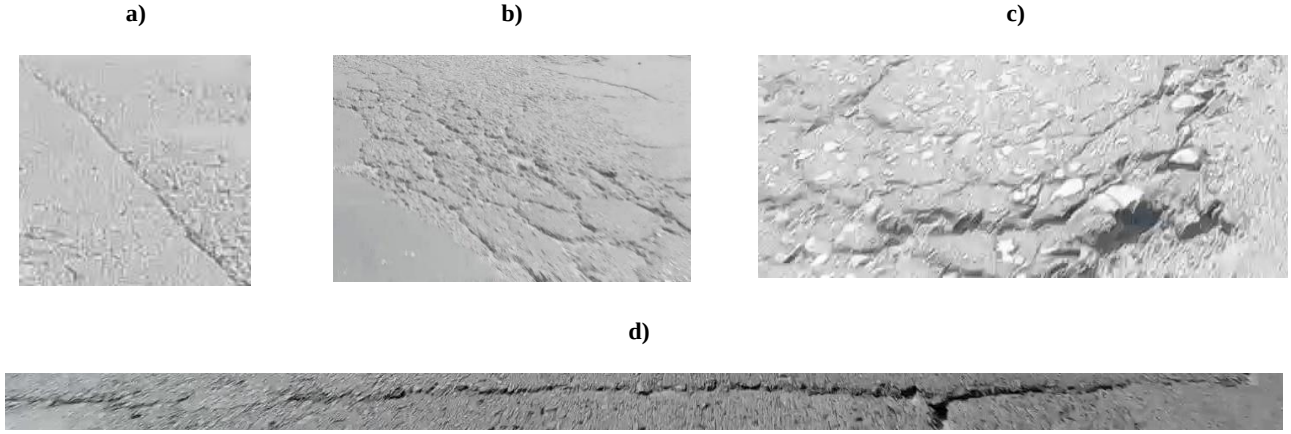

Fig. 12. Examples of classes: a) C01, b) C03, c) C04, d) C02

In addition to images with annotations, the dataset also includes images with no road defects present, which allows validation of the occurrence of False Positives. In addition, the dataset contains annotations specifying, additional information about a given image, such as whether there is a shadow in the image, painting on the road, outlandish - foreign element (e.g. sand, leaves, etc.), patch, milling of the road, occurrence of grain/binder defect and occurrence of manhole. The numbers of additional annotations are shown in Figure 13.

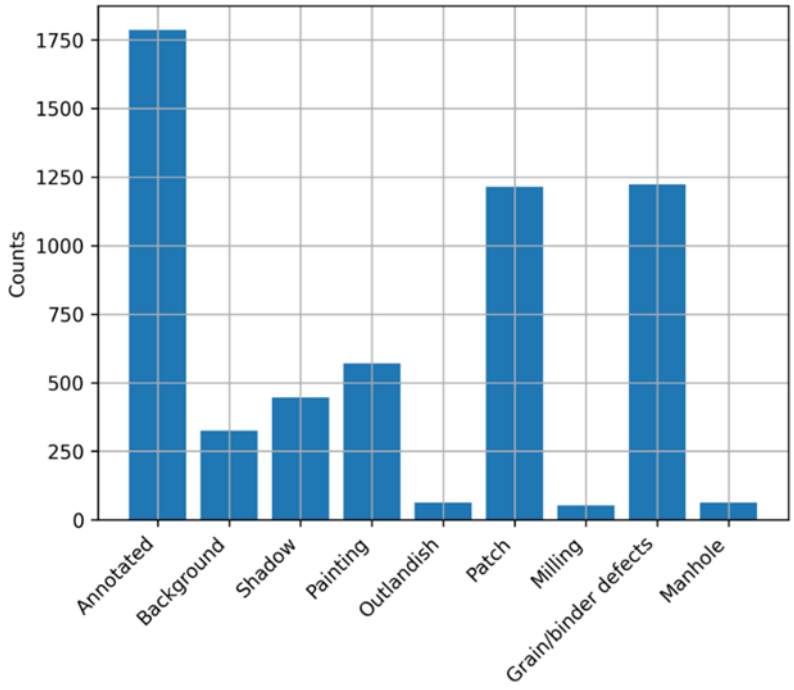

Table 13. Numbers of additional annotations in CoCGRCDD

The entire data set consists of 2110 images. To check the architectures, the best architecture from previous test - LANA-YOLO and the base model - YOLOv8n were selected. Both models were trained with pre-initialized weights from training on the CRDDC set. The number of training epochs was 300, of which mosaic augmentation was used for 250, and for another 50 the input data was augmented with horizontal flip, change in HSV value, etc. A comparison of model metrics is shown in Table 4.

Tab. 4. comparison of model metrics

Metric	**Yolov8**		**LANA-yolo**	
	Validation	**Train**	**Validation**	**Train**
mAP50	0.495	0.419	0.500	0.421
mAP50-95	0.240	0.180	0.242	0.182
Precision	0.518	0.510	0.520	0.515
Recall	0.451	0.433	0.458	0.432
F1-socre	0.480	0.460	0.480	0.460
TCPR* [GFLOPs]	8.1		6.8	

* TCPR - Theoretical computing power requirements

The LANA-YOLO model achieved better metrics on the validation set despite having fewer parameters. The difference between the models is small, amounting to 0.005 for the mAP50 metric, and only 0.002 for mAP50-95. Looking at the increase in model performance, such a change is insignificant, but considering the 16.05% reduction in the number of parameters, as well as the 14.34% lower inference time on the Jetson Xavier NX, it can be concluded that the developed model is better suited to tasks requiring real-time inference. An example comparison of the prediction results of the two models is shown in Figure 13.

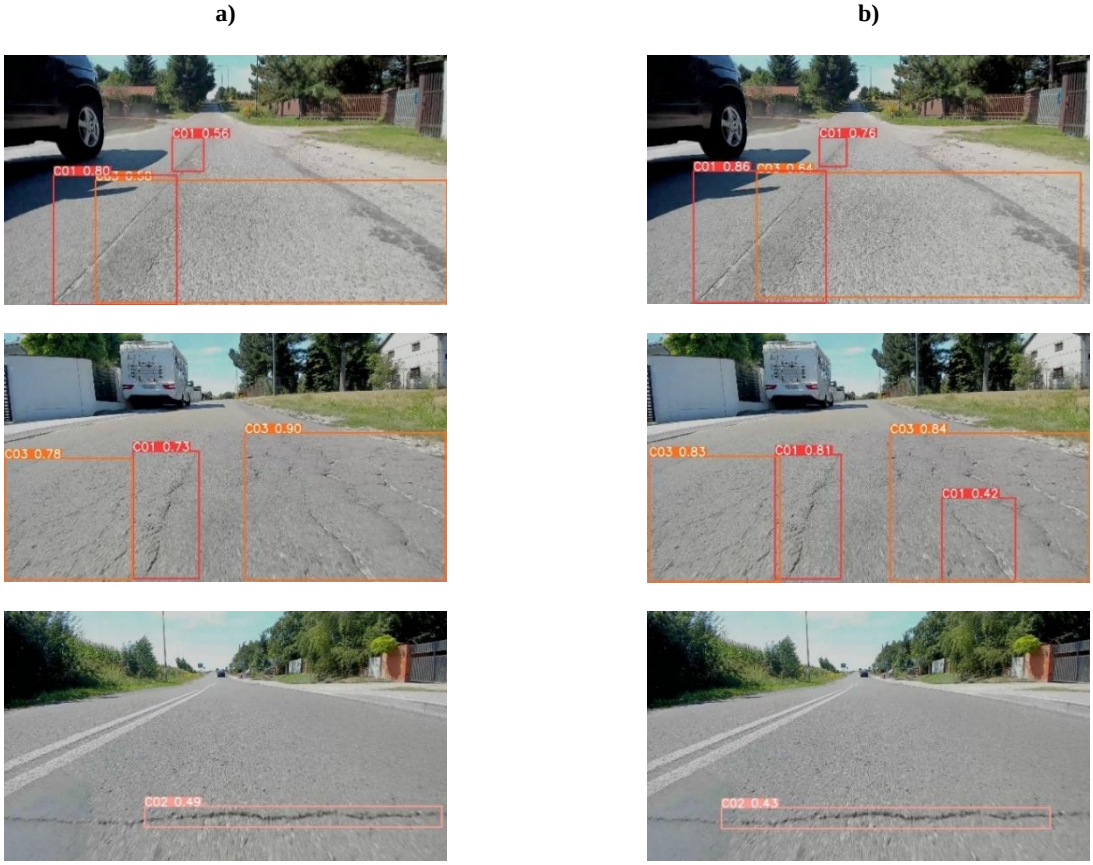

Fig. 13. Prediction results of: a) LANA-YOLO, b) YOLOv8n

On the basis of Figure 13, it can be observed that both models have similar efficiencies. In the case of the first row of images, the LANA-YOLO model shows less confidence for bounding boxes relative to the yolov8n model, but in the case of the second row of images, it can be noted that the YOLOv8 models are worse at delineating the occurrence of longitudinal cracks. Inside the bounding box defining the alligator crack, there is an indication of a longitudinal crack, which is not the expected result. For the last row of images, both models correctly indicated the presence of a transverse crack, but in both cases the entire area was not marked. Based on the study, it was concluded that the efficiency of the developed solution is higher than the base model while reducing the number of parameters and model inference time.

Saliency maps analysis using xGradCam was conducted to closely examine the differences in the decision-making process of the models (Fu et al., 2020). The maps were made from gradients, from the last C2f layers before head in the case of YOLOv8 and from the last convolution layers before head in the case of LANA-YOLO. Figure 14 shows example maps for yolov8 and LANA-YOLO.

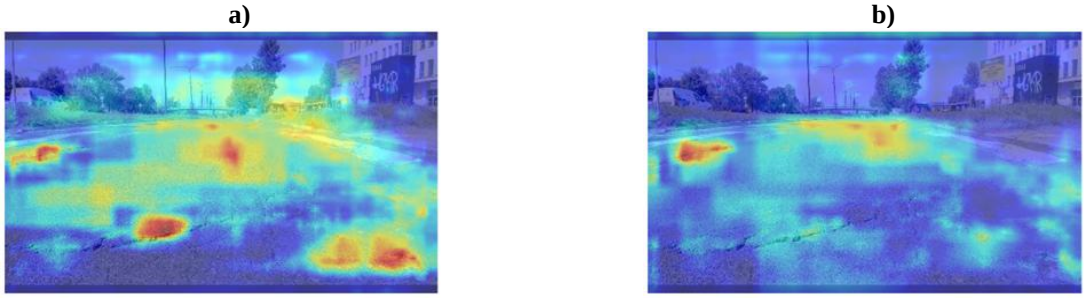

Fig. 14. Saliency maps for: a) LANA-YOLO, b) YOLOv8n

Analyzing the figure above, it can be seen that the YOLOv8 model tends to focus on low-level features, such as fine image details. While this allows for the detection of small and precise anomalies, it limits its ability to identify more extensive structures, such as large horizontal cracks, which require the integration of

information from wider areas of the image. In response to these limitations, the developed LANA-YOLO model was designed with the ability to efficiently process both low-level and high-level features. Through the use of advanced mechanisms such as LSK-A and enhanced network structures - AFPN with BIB, the model is able to accurately detect remote cracks while identifying cavities characterized by more extensive and complex structures.

4. CONCLUSIONS AND FUTURE WORKS

The developed architecture of the LANA-YOLO model is characterized by both higher accuracy relative to the other proven architectures, and has fewer parameters and lower theoretical computing power requirements, making its inference time on devices with limited computing power less than that of the base model. The use of LSK allowed an increase in accuracy, however, the use of this type of layer is associated with an increase in parameters. In the developed architecture, to reduce the number of parameters, as well as to increase the accuracy of the model, Neck was used in the form of AFPN with the application of the developed BIB block. The developed architecture achieved an mAP50 increase of 3pp over the base model on the CRDDC set while reducing parameters by 16.05%. For the developed PRCDD set, the LANA-YOLO model achieved mAP50 greater by 0.5pp. Tests conducted on various devices showed that the developed architecture allows for faster inference on various devices. In the case of the GPU, the model was converted to the ONNX format AND achieved an inference time that was 5.5ms faster than the base model. In the case of the raspberry pi 5, the model was also converted to ONNX format I achieved a time that was 2.27ms faster, while in the case of the Jetson Xavier NX, conversion to TRT format was used, where the model achieved a time that was 5.9ms faster.

The target solution is to create a system based on an artificial neural network model, a single-chip computer and a Global Position System receiver. The task of this system will be to infer on Edge and send information to the appropriate server regarding the location of given road surface defects. Such a solution will optimize the process of road asset management, which can help improve comfort and safety on the road.

Despite obtaining satisfactory results from the developed architecture, the whole solution needs further improvement in terms of architecture, auxiliary algorithms and hardware solutions. In further work, Global Shutter will be used, which will avoid the phenomenon of blurred images, in addition, the developed data set will be extended with additional images. Auxiliary algorithms such as Slicing Aided Hyper Inference - SAHI tested (Akyon et al., 2022) - will be tested.

Funding

This research was funded in whole or in part by National Science Centre, Poland 2024/08/X/ST6/00610.

Acknowledgements

For the purpose of Open Access, the author has applied a CC-BY public copyright license to any Author Accepted Manuscript (AAM) version arising from this submission.

Conflicts of Interest

The author declare no conflicts of interest.

REFERENCES

Akyon, F. C., Altinuc, S. O., & Temizel, A. (2022). Slicing aided hyper inference and fine-tuning for small object detection. *International Conference on Image Processing (ICIP)* (pp. 966–970). IEEE. https://doi.org/10.1109/ICIP46576.2022.9897990

Arya, D., Maeda, H., Ghosh, S. K., Toshniwal, D., Omata, H., Kashiyama, T., & Sekimoto, Y. (2022). Crowdsensing-based road damage detection challenge (CRDDC'2022). *2022 IEEE International Conference on Big Data (Big Data)* (pp. 6378–6386). IEEE. https://doi.org/10.1109/BIGDATA55660.2022.10021040

Arya, D., Maeda, H., Kumar Ghosh, S., Toshniwal, D., Omata, H., Kashiyama, T., & Sekimoto, Y. (2020). Global road damage detection: State-of-the-art solutions. *2020 IEEE International Conference on Big Data (Big Data)* (pp. 5533–5539). IEEE. https://doi.org/10.1109/BIGDATA50022.2020.9377790

Bai, R., Shen, F., Wang, M., Lu, J., & Zhang, Z. (2023). Improving detection capabilities of YOLOv8-n for small objects in remote sensing imagery: Towards better precision with simplified model complexity. https://doi.org/10.21203/RS.3.RS-3085871/V1

Bhattacharya, S., Jha, H., & Nanda, R. P. (2022). Application of IoT and artificial intelligence in road safety. *2022 International Conference on Interdisciplinary Research in Technology and Management (IRTM)* (pp. 1-6). IEEE. https://doi.org/10.1109/IRTM54583.2022.9791529

cvat-ai / cvat. (2025, March 21). Annotate better with CVAT, the industry-leading data engine for machine learning. Used and trusted by teams at any scale, for data of any scale. GitHub. Retrieved September 24, 2024 from https://github.com/cvat-ai/cvat

Fu, R., Hu, Q., Dong, X., Guo, Y., Gao, Y., & Li, B. (2020). Axiom-based Grad-CAM: Towards accurate visualization and explanation of CNNs. *ArXiv, abs/2008.02312v4*. https://arxiv.org/abs/2008.02312v4

Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., & Xu, C. (2020). GhostNet: More features from cheap operations. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, (pp. 1577–1586). IEEE. https://doi.org/10.1109/CVPR42600.2020.00165

Jakobsen, M. D., Glies Vincents Seeberg, K., Møller, M., Kines, P., Jørgensen, P., Malchow-Møller, L., Andersen, A. B., & Andersen, L. L. (2023). Influence of occupational risk factors for road traffic crashes among professional drivers: Systematic review. *Transport Reviews*, *43*(3), 533–563. https://doi.org/10.1080/01441647.2022.2132314

Jiang, Y. (2024). Road damage detection and classification using deep neural networks. *Discover Applied Sciences*, *6*, 421. https://doi.org/10.1007/s42452-024-06129-0

Li, D., Hu, J., Wang, C., Li, X., She, Q., Zhu, L., Zhang, T., & Chen, Q. (2021). Involution: Inverting the inherence of convolution for visual recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 12316–12325). IEEE. https://doi.org/10.1109/CVPR46437.2021.01214

Li, H., Li, J., Wei, H., Liu, Z., Zhan, Z., & Ren, Q. (2024). Slim-neck by GSConv: a lightweight-design for real-time detector architectures. *Journal of Real-Time Image Processing*, *21*, 62. https://doi.org/10.1007/S11554-024-01436-6

Li, Y., Hou, Q., Zheng, Z., Cheng, M. M., Yang, J., & Li, X. (2023). Large selective kernel network for remote sensing object detection. *IEEE International Conference on Computer Vision* (pp. 16748–16759). IEEE. https://doi.org/10.1109/ICCV51070.2023.01540

Liu, G., Reda, F. A., Shih, K. J., Wang, T.-C., Tao, A., & Catanzaro, B. (2018). Image inpainting for irregular holes using partial convolutions. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer Vision – ECCV 2018* (Vol. 11215, pp. 89–105). Springer International Publishing. https://doi.org/10.1007/978-3-030-01252-6_6

Liu, J., Zhang, S., Ma, Z., Zeng, Y., & Liu, X. (2023). A workpiece-dense scene object detection method based on improved YOLOv5. *Electronics*, *12*(13), 2966. https://doi.org/10.3390/ELECTRONICS12132966

Liu, Q., Huang, W., Duan, X., Wei, J., Hu, T., Yu, J., Huang, J., Liu, Q., Huang, W., Duan, X., Wei, J., Hu, T., Yu, J., & Huang, J. (2023). DSW-YOLOv8n: A new underwater target detection algorithm based on improved YOLOv8n. *Electronics*, *12*(18), 3892. https://doi.org/10.3390/ELECTRONICS12183892

Liu, S., Huang, D., & Wang, Y. (2019). Learning spatial fusion for single-shot object detection. *ArXiv*, *abs/1911.09516v2*. https://arxiv.org/abs/1911.09516v2

Ouyang, D., He, S., Zhang, G., Luo, M., Guo, H., Zhan, J., & Huang, Z. (2023). Efficient multi-scale attention module with cross-spatial learning. *IEEE International Conference on Acoustics, Speech and Signal Processing – Proceedings (ICASSP)* (pp. 1-5). IEEE. https://doi.org/10.1109/ICASSP49357.2023.10096516

Ranyal, E., Sadhu, A., & Jain, K. (2022). Road condition monitoring using smart sensing and artificial intelligence: A review. *Sensors*, *22*(8), 3044. https://doi.org/10.3390/S22083044

Tang, Z., Chamchong, R., & Pawara, P. (2023). A comparison of road damage detection based on YOLOv8. *International Conference on Machine Learning and Cybernetics* (pp. 223–228). IEEE. https://doi.org/10.1109/ICMLC58545.2023.10327993

Tomiło, P. (2024, October 1). CoCG Road Condition - Detection Dataset (CoCGRCDD). Mendeley Data. https://doi.org/10.17632/SNYYFKNW56.1

Tomiło, P., Oleszczuk, P., Laskowska, A., Wilczewska, W., & Gnapowski, E. (2024). Effect of architecture and inferencep of artificial neural network models in the detection task on energy demand. *Energies*, *17*(21), 5417. https://doi.org/10.3390/EN17215417

Wang, J., Meng, R., Huang, Y., Zhou, L., Huo, L., Qiao, Z., & Niu, C. (2024). Road defect detection based on improved YOLOv8s model. *Scientific Reports*, *14*, 16758. https://doi.org/10.1038/s41598-024-67953-3

Wang, X., Gao, H., Jia, Z., & Li, Z. (2023). BL-YOLOv8: An improved road defect detection model based on YOLOv8. *Sensors*, *23*(20), 8361. https://doi.org/10.3390/S23208361

Xing, Y., Han, X., Pan, X., An, D., Liu, W., & Bai, Y. (2024). EMG-YOLO: road crack detection algorithm for edge computing devices. *Frontiers in Neurorobotics*, *18*, 1423738. https://doi.org/10.3389/FNBOT.2024.1423738

Xu, H. (2022). FCD-YOLO: Improved YOLOv5 based on decoupled head and attention mechanism for defect detection on printed circuit board. *2022 2nd International Conference on Networking Systems of AI (INSAI)* (pp. 7–11). IEEE. https://doi.org/10.1109/INSAI56792.2022.00011

Yang, G., Lei, J., Zhu, Z., Cheng, S., Feng, Z., & Liang, R. (2023). AFPN: Asymptotic feature pyramid network for object detection. *IEEE International Conference on Systems, Man and Cybernetics* (pp. 2184–2189). IEEE. https://doi.org/10.1109/SMC53992.2023.10394415

Zhou, Y., & Yang, K. (2022). Exploring TensorRT to improve real-time inference for deep learning. *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)* (pp. 2011–2018). IEEE. https://doi.org/10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00299

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., & Dai, J. (2020). Deformable DETR: Deformable transformers for end-to-end object detection. *ArXiv, abs/2010.04159v4*. https://arxiv.org/abs/2010.04159v4