CIVIL AND ENVIRONMENTAL ENGINEERING REPORTS

E-ISSN 2450-8594

CEER 2025; 35 (1): 0177-0189
DOI: 10.59440/ceer/200378
Original Research Article

IMPLEMENTATION OF OBJECT DETECTION ALGORITHMS IN THE INVENTORY OF TECHNICAL INFRASTRUCTURE, CASE STUDY OF UTILITY POLES DETECTION

Mateusz CZYRZNIAK[1], Jacek RAPIŃSKI
[1] Department of Geodesy, Faculty of Geoengineering, University of Warmia and Mazury in Olsztyn, Poland

Abstract
Roads, bridges, railways, poles, and power lines are one of main elements of technical infrastructure. Their proper condition is essential for ensuring people, goods, and electricity transportation. Inventory is a crucial process for maintaining the health of technical infrastructure. Conducting inventory with traditional surveying techniques, such as Global Navigation Satellite Systems (GNSS) or trigonometry, is both time and money consuming. In recent years, modern remote sensing techniques like LiDAR (Light Detection and Ranging) and aerial and ground based imaging have been employed for inventory purposes. These methods enable data collection over large areas, reducing both the time and cost of measurement processes. Data from these devices is often processed in conjunction with artificial intelligence algorithms that assist in processing large data sets. This paper aims to introduce deep learning (DL) algorithms for inventory purposes using object detection with utility poles as a case study. The research was conducted using empirical data. A key element in training a DL model is the optimal selection of hyperparameters. During the study, ten models were trained, and their performance was compared based on selected metrics. For this study, a dataset of 1,736 original utility pole images was used. To augment the data, new images were created through rotation and mirroring. The best model achieved the following results: Precision = 98.82%, Recall = 97.29%, F1-score = 98.05%, mAP50 = 97.92%, mAP75 = 89.93%. The performance of the model gives a solid base for further implementation of DL object detection techniques for inventory of technical infrastructure.

Keywords: remote sensing, artificial intelligence, data processing, utility poles, deep learning

1. INTRODUCTION

The proper condition of technical infrastructure—such as roads, bridges, railways, poles, and power lines—is essential for the efficient transportation of goods, people, and electricity. To ensure the integrity of these elements, conducting periodic inventories is crucial [1]. Recently advanced

[1] Corresponding author: Mateusz Czyrzniak, University of Warmia and Mazury in Olsztyn, Faculty of Geoengineering, Department of Geodesy, Michała Oczapowskiego street 2, 10-719 Olsztyn, e-mail: mateusz.czyrzniak@uwm.edu.pl, phone: +48695092670

measurement techniques, like LiDAR (Light Detection And Ranging) [2] and imaging technologies [3], have been used for inventory purposes. These techniques enable data gathering across vast areas in a shorter time compared to classical surveying techniques like Global Navigation Satellite Systems (GNSS) [4]. The primary challenge of using modern measurement techniques is the time required for post-processing the collected data. To accelerate this process, Artificial Intelligence (AI) algorithms can be utilized [5].

AI algorithms can be used for tasks such as classification [6], segmentation [7] and object detection [8]. Among AI algorithms two subsets can be distinguished: machine learning and deep learning.

Machine learning (ML) "is the technique that improves system performance by learning from experience via computational methods" [9]. The idea of machine learning is to develop algorithms capable of making predictions on new data based on historical data. This subset of AI is used for tasks such as classification and segmentation and includes algorithms such as: Support Vector Machine (SVM) [10], Random Forest (RF) [11] and k-Nearest Neighbour (kNN) [12].

Deep learning (DL) is a specialized subset of machine learning. DL models are inspired by the structure and functionality of the brain. They use artificial neural networks, which are composed of layers of interconnected nodes called neurons. These networks can automatically learn hierarchical representations of data, enabling the model to extract features and patterns at multiple levels [13]. The example structure of artificial neural network is presented on Fig. 1. Each neuron from one layer is connected to all neurons from previous layer via connections. Each connection has assigned weight. The last layer is called the output layer and contains possible outputs. Each neuron (except those in input layer) makes calculations based on input values (from other neurons), weights and biases (Fig. 2). Then the activation function is applied to the output, which allows networks to learn non-linear patterns. Deep learning models are used for e.g. speech recognition, object detection, text analysis tasks, and linguistic models.

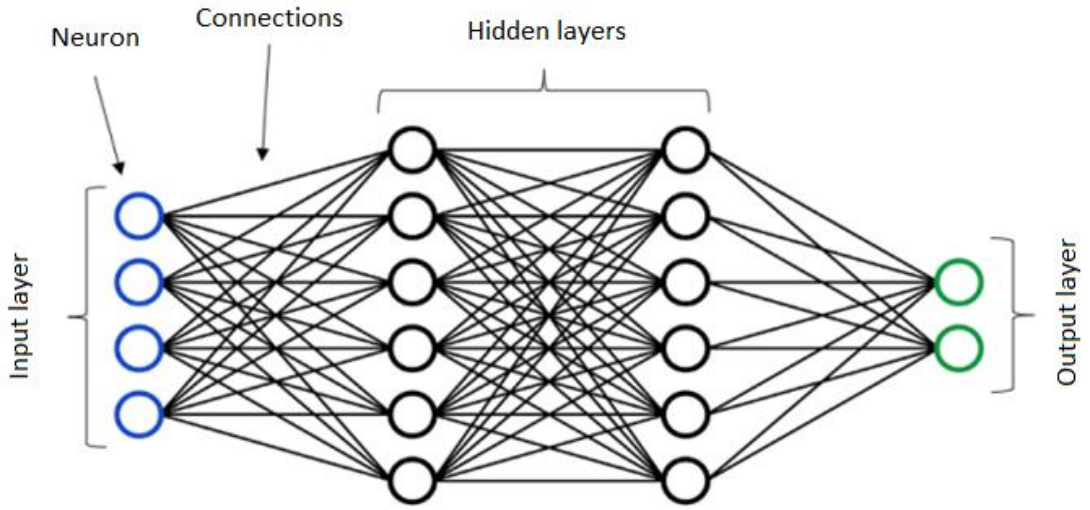


Fig. 1. Example structure of Artificial Neural Network [14]

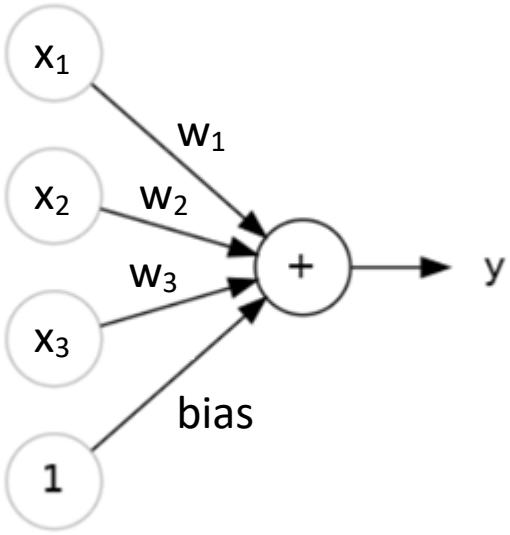


Fig. 2. Example of a close-up view of the neuron: x_1, x_2, x_3 – input values from previous layer; w_1, w_2, w_3 – weights assigned to the connections; y – calculated value of neuron [15]

In this research, the authors introduce an approach for detecting utility poles through DL object detection. The ultimate objective is to develop an algorithm capable of extracting specific information about poles from images, such as detection of pole cracking, insulator breakages and other failures. This paper presents the first step in creating that algorithm, focusing on the identification and extraction of utility poles from images.

The architecture of DL models can differ depending on the task they are designed for. Object detection models commonly use Convolutional Neural Networks (CNNs) [16]. CNNs are composed of multiple types of layers, among which three main types can be distinguished (Fig. 3):

- Convolutional layer: This layer applies filters (also called kernels) of a specified size to the input image. The filter slides over the image and performs calculations on pixels. As a result, a feature map is generated, representing the presence of specific features in the input image [16].
- Pooling layer: This layer is used to downsample feature maps. Similar to the convolutional layer it applies filters of a specified size over the feature map. This layer reduces the spatial dimensions (width and height) of the feature map [17];
- Fully-connected layer: This layer connects each neuron from one layer to every neuron in the next layer. It enables the model to perform classification tasks based on features extracted by the preceding layers [18].

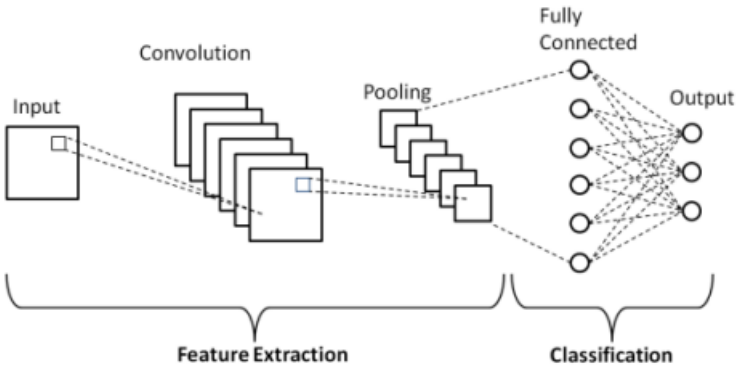


Fig. 3. Convolutional Neural Network [19]

2. METHODOLOGY

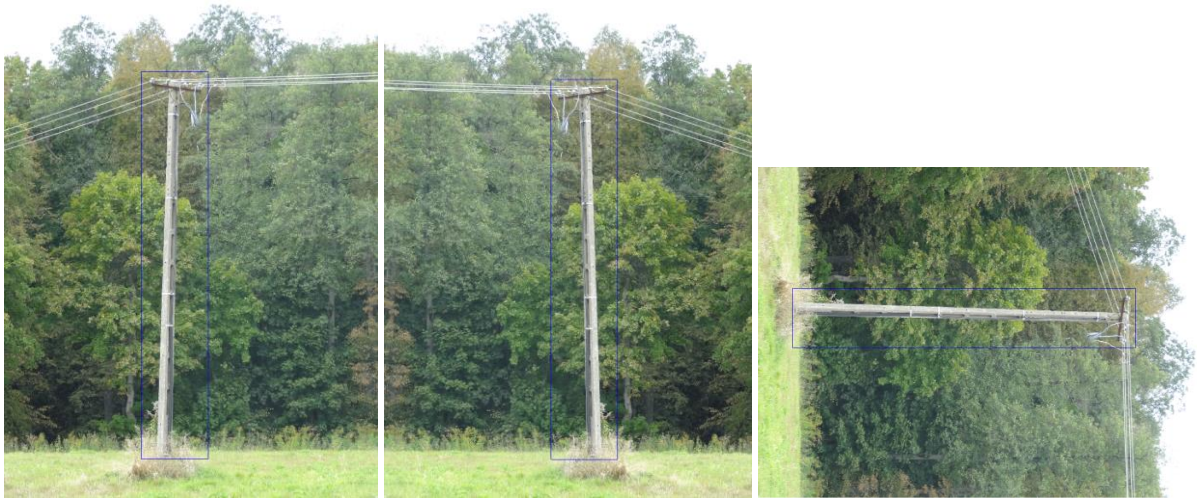
For utility object detection the YOLO (You Only Look Once) model [20] was selected. Number of images collected during inventory purposes can reach substantial sizes. That's why the chosen model must be fast and accurate. The YOLO model is known for its short detection time and is widely used for a wide range of applications, including real-time object detection. For the purpose of utility poles detection version 8 of the YOLO model was employed [21]. This version offers a well-balanced trade-off between speed and performance. This section is divided into three subsections: data collection, model training and model evaluation.

2.1. Data collection

Images of utility poles were collected using ground-based imaging techniques, resulting in a total of 1,736 original images. Images were collected using three cameras: the SONY DSC-HX10V, Xiaomi Redmi 4X, and Xiaomi MI 8 Lite, with image resolutions of 2592x1944, 3210x4160, and 3024x4032, respectively. Photos were taken under various lighting conditions, from different angles, and from different positions. To augment the dataset, additional images were generated by applying rotation and mirroring. Rotation was applied to both the original and mirrored images at 90°, 180°, and 270° angles (Fig. 4). To reduce the model's susceptibility to false positives, 4,964 random images from the Common Objects in Context (COCO) dataset [22] (a large-scale dataset used for object detection, segmentation, and captioning), which did not contain utility poles, were added to the collected data (Fig. 5). Ultimately, the dataset comprised 18,849 images. This dataset was then divided into three subsets: training, validation, and test datasets (Table 1). All variations of the same image were assigned to the same subset to prevent the model from learning the patterns.

Table 1. Collected data

	Test	**Train**	**Val**	**SUM**
Images in dataset	2536	14529	1784	18849
Utility poles images	1536	10968	1384	13888
Coco dataset images	1000	3561	400	4961



Original image Mirrored image Rotated image

Fig. 4. Labeled images

Fig 5. Example images from COCO dataset

2.2. Model training

A key factor in training a deep learning (DL) model is the optimal selection of hyperparameters. Hyperparameters are parameters that define the model's structure and govern how the model is trained [23]. Hyperparameters used in this study are listed and explained below:

- Number of epochs [24]: Specifies the number of times the entire training dataset is passed through the model during training;
- Optimization algorithm [25]: An algorithm used to minimize the model's loss function by updating the weights in order to improve models performance. The loss function measures the error between predicted and true values.
- Learning rate [26]: Controls the step size of the optimization algorithm. A small value slows the training process and risks of stucking in local minima. A large value can cause the model to overshoot the minimum.

- Cosine learning rate scheduler [27]: Adjusts the learning rate during the training process. A cosine learning rate scheduler gradually reduces the learning rate following a cosine curve during training.
- Early stopping [28]: Halts the training process when the model's performance stops improving.
- Batch size [29]: Refers to the number of training samples processed together in one iteration during model training.

In this study, 10 models with varying hyperparameter configurations were trained (Table 2). The researchers focused on the following hyperparameters:

- Optimization algorithm,
- Learning rate,
- Cosine learning rate scheduler.

The number of epochs was set to 300, with early stopping applied after 50 epochs if no significant progress was observed. The batch size was set to 100, which was optimal for the hardware used (Graphics processor: NVIDIA GeForce RTX 3090). For all remaining hyperparameters, default values were applied. Two models were trained with automatically selected hyperparameters (marked as Auto* in Table 2), with and without the use of the cosine learning rate scheduler. The variations in hyperparameters were as follows:

- Optimization algorithm: Stochastic Gradient Descent (SGD) [30] computes the gradient and updates the weights using single randomly selected training example or a small batch, Adaptive Moment Estimation with Weight Decay (AdamW) [31] utilizes adaptive learning rates derived from the first and second moments of gradients. Additionally it incorporates weight decay, which adds penalty to large weights to prevent overfitting;
- Learning rate: 0.01, 0.001;
- Cosine learning rate scheduler: True, False.

Table 2. Trained models

Model	Optimizer	Learning rate	Cosine learning rate scheduler	Training time [h]
Train31	Auto*	Auto*	False	6
Train32	Auto*	Auto*	True	8
Train33	SGD	0.01	False	10
Train34	SGD	0.01	True	9
Train35	SGD	0.001	False	5
Train36	SGD	0.001	True	5
Train37	AdamW	0.01	False	6
Train38	AdamW	0.01	True	7
Train39	AdamW	0.001	False	6
Train40	AdamW	0.001	True	10

***Auto**:
optimizer: SGD, lr=0,01, momentum=0,9 with parameter groups 57 weight(decay=0,0), 64 weight(decay=0,00078125), 63 bias(decay=0,0).

2.3. Model evaluation

Performance of the trained models was evaluated based on four metrics commonly used in DL: Precision, Recall, F1-score and mean Average Precision (mAP) with Intersection over Union (IoU) [32] over 50 (mAP50) and 75 (mAP75) percentage:

- Precision (2.1) – the percentage of correct positive predictions from all positive predictions made by the model [33].

$$Precision = \frac{T_P}{T_P + F_P} \tag{2.1}$$

Where: T_P – True Positive, F_P – False Positive

- Recall (2.2) – the percentage of correct positive predictions from all predictions that the model should have made [33].

$$Recall = \frac{T_P}{T_P + F_N} \tag{2.2}$$

Where: T_P – True Positive, F_N – False Negative

- F1-score (2.3) – combines precision and recall to correctly estimate the model's trustworthiness [34].

$$F1 - score = 2\frac{Precision * Recall}{Precision + Recall} \tag{2.3}$$

- Mean Average Precision – the mean of Average Precision (AP) (2.4) across all classes [35]. AP is the area under the Precision-Recall (PR) curve [36].

$$AP = \int_0^1 Precision(Recall) dRecall \tag{2.4}$$

3. RESULTS

All the models achieved very good results in terms of selected evaluation metrics. In terms of Precision (Fig. 6), Recall (Fig. 7) and F1-score (Fig. 8) all models achieved values above 95%. The different colours in Figure 6, Figure 7 and Figure 8 correspond to the different optimisation algorithms. The highest values were achieved by model "Train32" with scores of Precision, Recall and F1-score: 98.82%, 97.29%, 98.05% respectively. For mAP50 the highest value was 98.40% achieved by model "Train39", for mAP75, model "Train31" reached score of 91.93%. All metrics are shown in Table 3. Sample results of pole detection on the test dataset are shown in Fig. 9. Each image in Fig. 9 from a) to h) represents detected bounding box around the utility pole together with a label and confidence score in top left corner of the bounding box. Images a) and b) represent normal images of utility poles. Images c) and d) shows utility poles rotated by approximately 90°. Image e) shows a utility pole at an oblique angle. Images f), g) and h) shows utility poles rotated by approximately 270°.

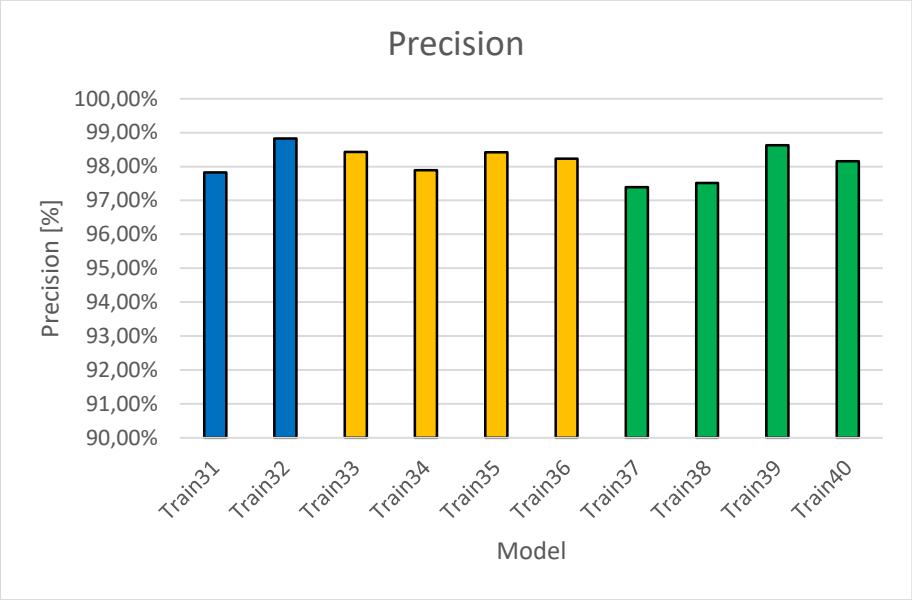


Fig. 6. Precision values of trained models

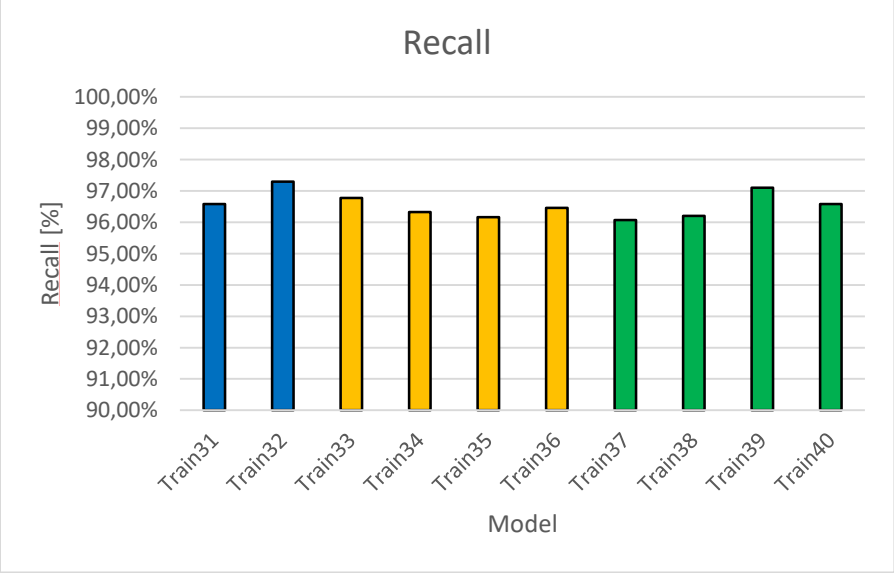


Fig. 7. Recall values of trained models

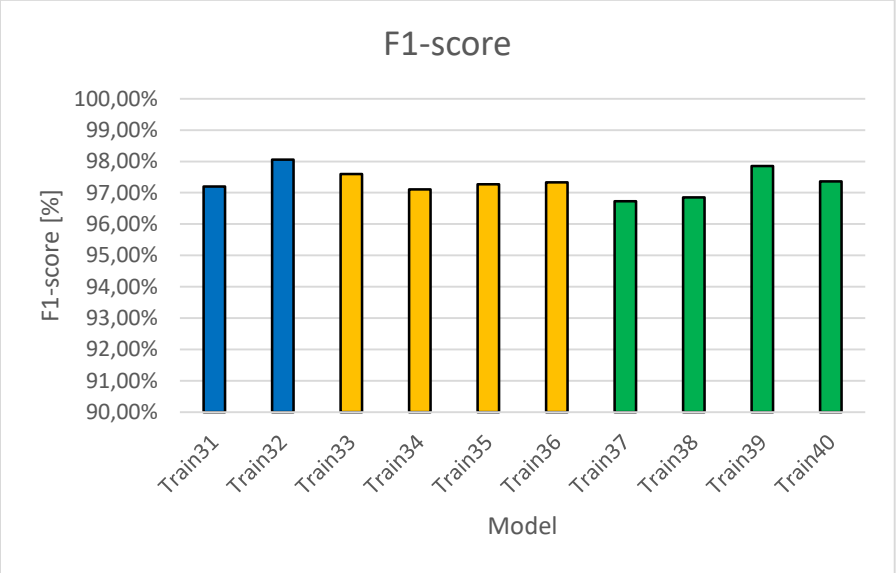


Fig. 8. F1-scores of trained models

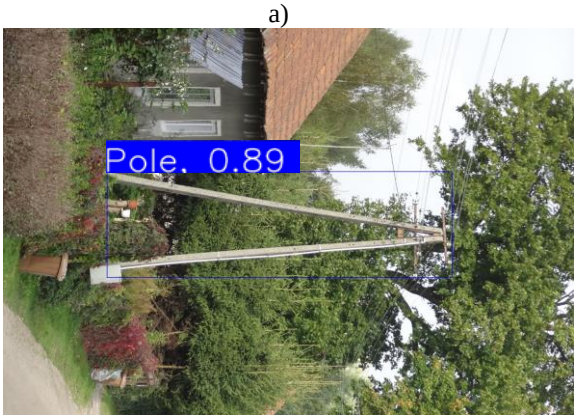
Table 3. Values of evaluation metrics of trained models

	Train31	**Train32**	**Train33**	**Train34**	**Train35**	**Train36**	**Train37**	**Train38**	**Train39**	**Train40**
Precision [%]	97.83	98.82	98.43	97.89	98.42	98.23	97.39	97.52	98.63	98.16
Recall [%]	96.59	97.29	96.78	96.33	96.16	96.46	96.07	96.20	97.10	96.59
F1 [%]	97.20	98.05	97.60	97.10	97.27	97.33	96.72	96.85	97.86	97.36
mAP50 [%]	97.08	97.92	97.41	97.44	97.79	97.77	97.34	97.73	98.41	97.80
mAP75 [%]	91.93	89.93	88.90	89.09	88.35	88.95	91.04	90.23	90.91	89.76

Analyzing Table 3, one can notice that models present similar results varying up to 1-3% points in terms of individual metric. The model “Train32” with highest values in terms of Precision, Recall, and F1-score was trained with automatically selected hyperparameters and using cosine learning rate scheduler (Table 2). In terms of mAP50 it was surpassed only by the model “Train39”. In terms of mAP75 models: “Train31”, “Train37”, “Train38”, and “Train39” achieved higher scores.

a)

b)



c)

d)

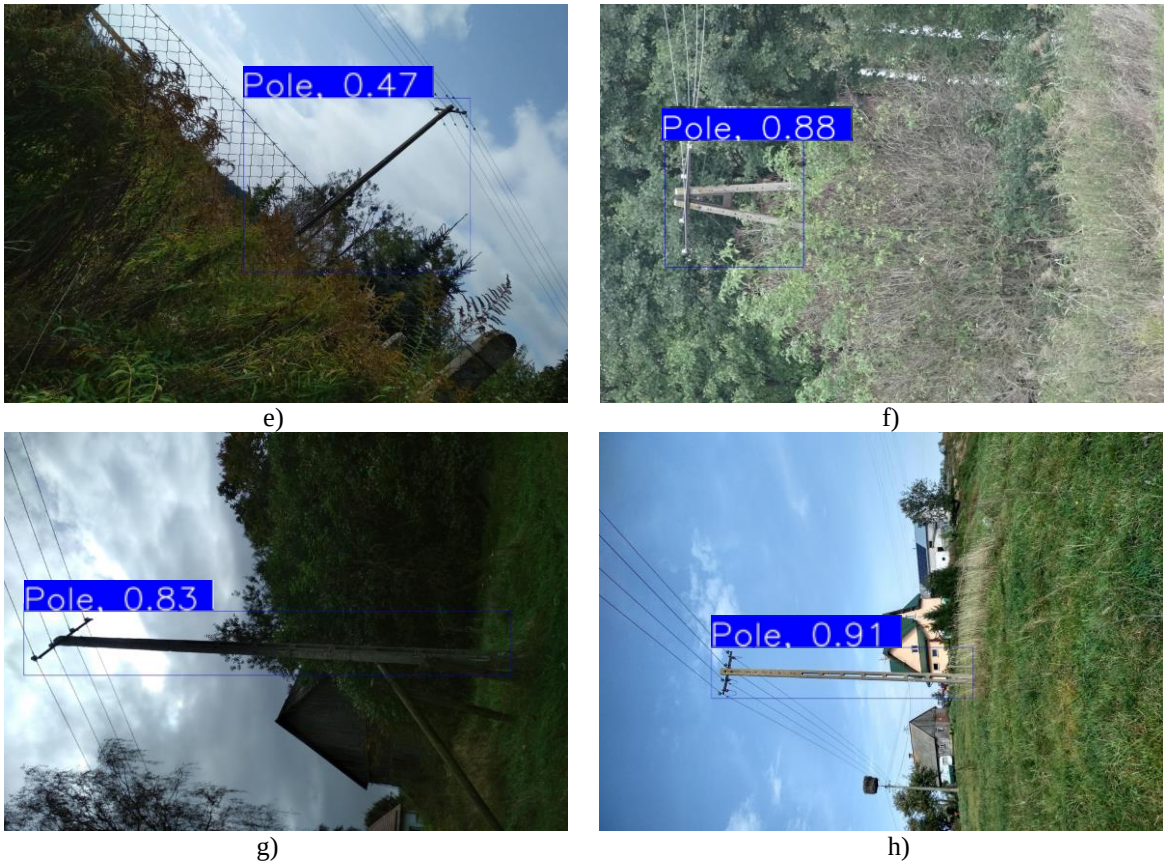


Fig. 9. Sample results of the utility poles detection on the test dataset: the label contains name of object and confidence score of detection. Subfigures contain a detection of: a) and b) utility pole, c) utility pole rotated 90°, d) utility pole rotated 90°, e) utility pole in oblique angle, f), g) and h) utility pole rotated 270°

4. CONCLUSIONS

Processing data acquired through modern measurement techniques can be very time-consuming. The results obtained in this study clearly demonstrate that AI models can be employed to process the data, significantly saving both time and money. However, depending on the specific purpose of the inventory, different metrics should be prioritized. If the objective is identifying poles, a good leading metric would be the F1-score. Conversely, if the goal is to extract the entire object from the image, the mAP with satisfying IoU threshold is a more appropriate metric.

In this case study on utility pole detection, the model "Train32" achieved the best results in terms of Precision, Recall, and F1-score, with values of 98.82%, 97.29%, and 98.05%, respectively. For the mAP50 metric, the highest value was obtained by the model "Train39" at 98.41%. The highest mAP75 value, 91.93%, was achieved by the model "Train31". These results are highly promising. Depending on the possible application of developed models, different models should be used: identification of poles – model “Train32”, extraction of the entire objects – model "Train31" or "Train39".

The performance of the models can vary when applied to images with different resolutions. In this situation images can be rescaled or the model can be fine-tuned using additional data.

In future research, the trained model will be used to extract utility poles from images for further analysis, such as insulators or crack detection. These elements are small and may be difficult to detect in a low-resolution images or in images taken from a distance. To augment collected data integrating other sensors such as LiDAR and RADAR (Radio Detecting and Ranging) can be considered. However, this approach creates new challenges in terms of processing and integration of data from multiple sources.

REFERENCES

1. Sharifisoraki, Z et al. 2023. Monitoring Critical Infrastructure Using 3D LiDAR Point Clouds. *Institute of Electrical and Electronics Engineers Inc.*, **11**, 314-336.
2. Soilán, M et al. 2019. Review of laser scanning technologies and their applications for road and railway infrastructure monitoring. *MDPI Multidisciplinary Digital Publishing Institute*, **4(4)**, 58.
3. Mohan, A and Poobal, S 2018. Crack detection using image processing: A critical review and analysis. *Alexandria Engineering Journal* **57(2)**, 787-798.
4. Schlögl, M et al. 2022. Remote Sensing Techniques for Bridge Deformation Monitoring at Millimetric Scale: Investigating the Potential of Satellite Radar Interferometry, Airborne Laser Scanning and Ground-Based Mobile Laser Scanning. *PFG - Journal of Photogrammetry, Remote Sensing and Geoinformation Science* **90(4)**, 391-411.
5. Pepe, M et al. 2022. Data for 3D reconstruction and point cloud classification using machine learning in cultural heritage environment. Data Brief 42, 108250.
6. Grilli, E, Menna, F and Remondino, F 2017. A Review of Point Clouds Segmentation and Classification Algorithms. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **XLII-2/W3**, 339–344.
7. Balado, J et al. 2019. Road environment semantic segmentation with deep learning from mls point cloud data. *Sensors (Switzerland)* **19(16)**, 3466.
8. Fukano, K and Masuda, H 2015. Detection and Classification of Pole-Like Objects from Mobile Mapping Data. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **II-3/W5**, 57–64.
9. Zhi-Hua, Z 2021. *Machine_Learning*. Singapore: Springer Nature.
10. Awad, M and Khanna, R 2015. Support Vector Machines for Classification. In: Awad, M and Khanna, R (ed) *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers* Berkeley, CA: Apress, 39-66.
11. Parmar, A, Katariya, R and Patel, V 2019. *A Review on Random Forest: An Ensemble Classifier*. International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI), Springer International Publishing, 758-763.
12. Abu Alfeilat, HA et al. 2019. Effects of Distance Measure Choice on K-Nearest Neighbor Classifier Performance: A Review. *Big Data* **7(4)**, 221-248.
13. LeCun, Y, Bengio, Y and Hinton, G 2015. Deep learning. *Nature* **521(7553)**, 436-444.
14. Wu, Y and Feng, J. 2018. Development and Application of Artificial Neural Network. *Wirel Pers Commun* **102(2)**, 1645-1656.
15. Agatonovic-Kustrin, S and Beresford, R 2000. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. *J Pharm Biomed Anal* **22(5)**, 717-727.
16. Gu, J et al. 2018. Recent advances in convolutional neural networks. *Pattern Recognit* **77**, 354-377.
17. Sun, M et al. 2017. Learning Pooling for Convolutional Neural Network. *Neurocomputing* **224**, 96-104.

18. Basha, SHS et al. 2020. Impact of fully connected layers on performance of convolutional neural networks for image classification. *Neurocomputing* **378**, 112-119.
19. Phung and Rhee 2019. A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets. *Applied Sciences* **9**, 4500.
20. Redmon, J et al. 2016. You Only Look Once: Unified, Real-Time Object Detection. *ArXiv*.
21. Sohan, M, Sai Ram, T and Rami Reddy CHV 2024. A Review on YOLOv8 and Its Advancements. In: Jacob, IJ, Piramuthu, S and Falkowski-Gilski, P (ed) *Data Intelligence and Cognitive Informatics*. Singapore: Springer Nature Singapore, 529-545.
22. Lin, TY et al. 2015. Microsoft COCO: Common Objects in Context. *ArXiv*.
23. Yang, L and Shami, A 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* **415**, 295-316.
24. Chilimbi, T et al. 2014. *Project Adam: Building an Efficient and Scalable Deep Learning Training System.* 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14), , CO: USENIX Association, Broomfield, 571-582.
25. Zhang, J 2019. Gradient Descent based Optimization Algorithms for Deep Learning Models Training. *ArXiv*.
26. Wu, Y et al. 2019. *Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks*. IEEE International Conference on Big Data (Big Data), 1971-1980.
27. Yang, J and Wang, F 2020. Auto-Ensemble: An Adaptive Learning Rate Scheduling Based Deep Learning Model Ensembling. *IEEE Access*, **8**, 217499–217509.
28. Bai, Y et al. 2021. Understanding and Improving Early Stopping for Learning with Noisy Labels. In: Ranzato, M et al. (ed) *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 24392-24403.
29. Kandel and Castelli, M 2020. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, **6(4)**, 312-315.
30. Xing, C et al. 2018. A Walk with SGD. *ArXiv*.
31. Zhou, P 2024. Towards Understanding Convergence and Generalization of AdamW. *IEEE Trans Pattern Anal Mach Intell*, **46(9)**, 6486-6493.
32. Rezatofighi, H et al. 2019. *Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression.* Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 658-666.
33. Powers, DMW and Ailab Evaluation: From Precision, Recall And F-Measure To Roc, Informedness, Markedness & Correlation.
34. Chicco, D and Jurman, G 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics* **21(1)**, 6.
35. Henderson, P and Ferrari, V 2017. End-to-end training of object class detectors for mean average precision. *ArXiv*.
36. Padilla, R, Netto, SL and Silva, EAB 2020. *A Survey on Performance Metrics for Object-Detection Algorithms*. International Conference on Systems, Signals and Image Processing (IWSSIP), 237-242.