

Benchmarking Generations of You Only Look Once Architectures for Detection of Defective and Normal Long Rod Insulators

Gergő Bendegúz Békési

Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Magyar tudósok krt. 2. (Bldg. Q.), Budapest, H-1117, Hungary.

Contributing authors: bekesigergobendeguz@edu.bme.hu;

Abstract

The effective infrastructure monitoring is a priority in all technical fields in this century. In high-voltage transmission networks the line inspection is one such task. The fault detection of insulators is very important and object detection algorithms can handle this problem. A comparison of You Only Look Once architectures is presented in this work. The different subtypes of the last three generations (v3, v4, v5) are compared in terms of losses, precisions, recalls and mean average precisions. It is found out that YOLOv5 subtypes are the most suitable to perform this computer vision task considering both accuracy and speed.

Keywords: YOLOv3, YOLOv4, YOLOv5, insulator detection, insulator fault detection

1 Introduction, Positioning

The reliability of the transmission network is of paramount importance, especially due to electrification and the increase in global energy demand. The failure of insulators in this system can result in serious economic consequences. Therefore, special attention should be paid to them. Insulators of different types and made of different materials have different failure mechanisms. This paper focuses on the detection of strain type porcelain long rod insulators that are typically used in high voltage transmission lines as overhead insulation. The tension they accommodate is influenced by temperature and wind load, and can be influenced by ice load. Long rod insulators are considered to be unpuncturable, but they can crack or brake under thermal stress ([Macey et al, 2004](#)). To help

the detection of these faults of the power systems in time, unmanned aerial vehicles (UAV) and computer vision algorithms can be applied.

Object detection is one of the most fundamental challenges of computer vision. There are basically three approaches that can be used to cope with this task. The oldest methods are the traditional solutions like Viola-Jones ([Viola and Jones, 2001](#)) or Deformable Part-based Model algorithm ([Felzenszwalb et al, 2008](#)), which are nowhere near as robust as their deep learning successors. Methods using deep neural networks are basically classified into two groups ([Zou et al, 2019](#); [Zou, 2019](#)). These two are the One-stage and Two-stage or in other words the You Only Look Once (YOLO) / Single Shot Detector (SSD) type and region-based groups. It is shown in several works that YOLO achieves good trade-off between

speed and accuracy as opposed to region-based methods which are not fast enough. Lee and Kim (2020) draw this conclusion in connection with airplane detection. Kim et al (2020) makes a similar finding in the case of vehicle detection. Furthermore, Li et al (2020) and Sumit et al (2020) outline this when detecting agricultural greenhouses and human figures. Thus, it can be assumed that a One-stage architecture may be a better choice for real-time recordings made by UAVs.

Among these methods, due to its good performance, the most popular solution is to use YOLO architectures. In addition to the research fields already listed, it has many other applications. Loey et al (2021) presents medical face mask detection with YOLO v2 and achieves average precision percentage of 81 % with Adam optimizer and Res-Net-50 feature extractor. Ma et al (2020) performs collapsed building detection in post-earthquake high-resolution remote sensing images with improved YOLOv3. The presented model reaches target precision of 90.89 % and speed of 29.23 frame per second. Fu et al (2021) achieves higher detection accuracy for small targets, multiple targets and overlapping targets with YOLOv4 in the field of marine target detection that plays an important role in sea area management, ship collision avoidance and sea surface monitoring. The mean average precision (mAP_{50}) of the developed algorithm is 84.06 %. YOLOv5 model reaches precision values between 86.5 % and 86.8 % in plant disease recognition (Chen et al, 2022b). The scope of this study is powdery mildew and anthracnose detection, that are the most typical diseases of rubber trees. Along with the few examples highlighted, there are many other applications based on versions of YOLO model (Yu and Choi, 2022; Zhang et al, 2021; Xu et al, 2021; Shi et al, 2021; Avazov et al, 2022; Celik, 2022; Ardhiyanto et al, 2022; Shao et al, 2022; Zhou et al, 2021; Chen et al, 2021; Junos et al, 2022; Wang et al, 2020b; Gai et al, 2021; Karaci, 2022; Al-qaness et al, 2021; Xianbao et al, 2021; Hu et al, 2021; Cheng et al, 2021; Zhao et al, 2022; Dewi et al, 2022; Wageeh et al, 2021; Kumar et al, 2021; Chen and Miao, 2020).

In the field of insulator and defective insulator detection, the use of YOLO methods is not exclusive, there are other algorithms as well. Among the works from 10-12 years ago, there are solutions made with traditional image processing

algorithms. Examples include the use of the Hough Transform (Zhang et al, 2010) or the Support Vector Machine (SVM) Algorithm (Li et al, 2012). Today, deep-learning algorithms are used for these tasks due to their robustness (Van et al, 2018). There are also One-stage and Two-stage solutions. Among the region-based methods Zhang et al (2022) uses Faster R-CNN to locate insulators and extract them from the image. After that a segmentation is also applied in this work, which outlines that YOLO is a faster but less accurate method. Hu et al (2022) reaches an 1.63 % increase in mAP_{50} with replacing the VGG16 backbone of the network with Resnet50 and adding a channel attention module to the architecture. Zhao et al (2019) improves the anchor generation method of the R-CNN network and to perform an 81.8 % average precision which would be 27.81 % less without the improvement. In addition to those listed here, other region-based solutions can be found (Tomaszewski et al, 2020; Ma et al, 2017; Wang et al, 2020c). The weak point of these approaches is their inability to run in real time. Given the limited battery capacity of UAVs and the length of lines in transmission networks, it is definitely desirable for UAVs not to spend a lot of time on an insulator. But it is undeniable, that these are currently the most accurate algorithms for object detection.

One-stage insulator detection can be separated into YOLO and SSD methods. Typically, SSD methods provide the fastest solution, but can be very inaccurate when detecting small objects. The issue of small objects is particularly important, as in this paper an UAV video recording based application is presented. It should also be emphasized that due to strong electromagnetic fields, UAVs cannot go very close to the insulators easily. Thus, one of the YOLO architectures may be more suitable for this application. Especially since significant progress has been made since YOLOv3 to detect small objects directly (Liu et al, 2020). Over the past year or two, several researchers have addressed this topic. Chen et al (2022a) proposes a YOLOv4 based method especially for insulator detection. This model called INSU-YOLO has a bit more parameters than the original YOLOv4 and performs better by 4.59 %, although it has a bit lower frame per second (FPS) value. Liu et al (2021c) makes an improvement on the YOLOv3 architecture and achieves good performance for

insulator fault detection in aerial images with diverse backgrounds. The same research group (Liu et al, 2021b) in another work increases the feature reuse in the low-resolution feature layers of YOLOv3 and YOLOv4 architectures and also improves the loss function. Liu et al (2021a) does similar feature reuse with YOLOv2 and YOLOv3 based architectures and achieves 94.47 % precision with YOLOv3-dense. YOLOv5 is used in many new works as the newest version of YOLO generations. Feng et al (2021) tries four different versions of YOLOv5 and achieves 95.5 % mAP_{50} with it. Gao et al (2021) improves the feature extraction by establishing a certain spatial relationship between the residual transformation and the rotation operation and reaches 93.4 % mAP_{50} . As well as others, Feng and Jiang (2021) use different versions of YOLOv5, but in case of recognizing station post insulators.

Some of the applications listed here only perform insulator detection, others detect faults as well. The ultimate goal of all research is the same: to help the work of transmission system operators by filtering out individual insulator faults semi-automatically or fully automatically. YOLO architectures can be a good solution to this problem, and several versions of them are used in several studies. However, the benchmarking of different versions of the YOLOv3, YOLOv4, and YOLOv5 architectures is still poorly published in the field of insulator detection.

The main contribution of this work is the comparison of architectures YOLOv5n (bib, 2020b), YOLOv5s (bib, 2020b), YOLOv5m (bib, 2020b), YOLOv5l (bib, 2020b), YOLOv5x (bib, 2020b), YOLOv5n6 (bib, 2020b), YOLOv5s6 (bib, 2020b), YOLOv5m6 (bib, 2020b), YOLOv5l6 (bib, 2020b), YOLOv4 (bib, 2020c), YOLOv4-tiny (bib, 2020c), YOLOv4-csp (bib, 2020c), YOLOv3 (bib, 2020a), YOLOv3-tiny (bib, 2020a) and YOLOv3-spp (bib, 2020a). The main aspects of comparison are Precision, Recall, mAP_{50} , box loss, object loss and class loss.

The rest of the paper is organized as follows. Section 2 introduces the methods and data used for the work. Evaluation and Results are presented and discussed in Section 3, and finally, conclusions are drawn in Section 4.

2 Methods and Data

In this section, the deep-learning frameworks are presented. The first subsection concerns the basics of the YOLO algorithm. The newer versions and types of YOLO are presented in the second, third and fourth part, and the data management is described in the fifth part.

2.1 The basics of the YOLO algorithm

Here, a brief overview of the basic concepts of the YOLO algorithm is presented. This is how YOLO works up to version v2. The basic operation of YOLO is shown in Figure 1.

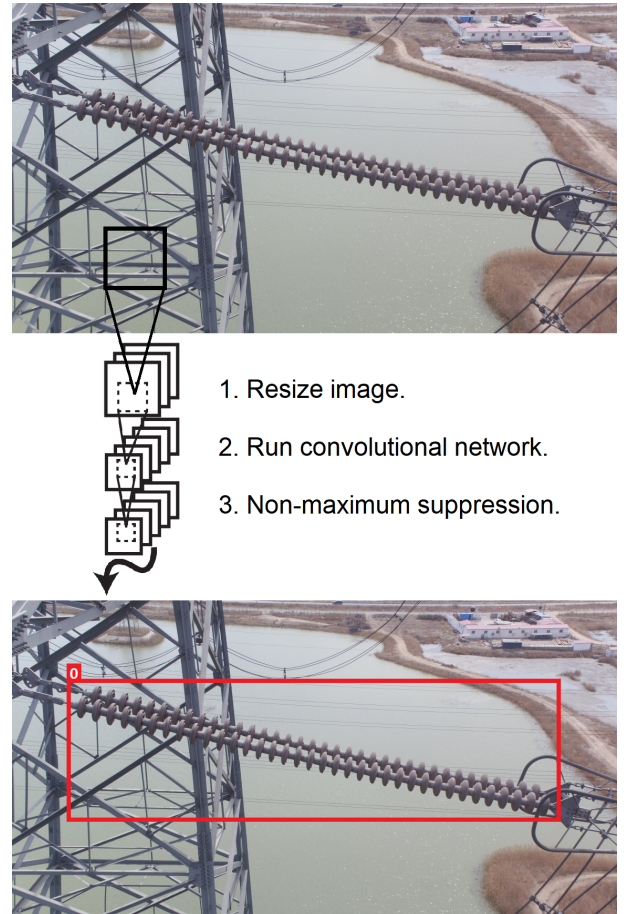


Fig. 1 The YOLO Detection System (Redmon et al, 2016)

The essence of the YOLO algorithm is that it does not use region proposals, like say RCNN, but generates a lot of bounding box outputs and

selects the appropriate ones. YOLO is a fully convolutional network (FCN) architecture that compresses an image into an $S \cdot S$ layout with N dimension grid cells thanks to its different stride values. Bounding boxes can be estimated from these grid cells. If B bounding boxes from each grid cell are estimated, a total of $S \cdot S \cdot B$ bounding boxes are available by the end of the process. Importantly, each object will be estimated from the grid cell where its center is after the stride operations. The so-called non-maximum suppression is also used. If a cell results several significantly overlapping bounding boxes associated with an object, the highest confidence bounding box has to be chosen. The bounding box estimation process is as follows. B bounding boxes from each grid cell are estimated with the same parameters. This gives $(5 + NC) \cdot B$ outputs from each cell. NC is the number of classes, and the other 5 parameters are x , y , w , h , and *confidence*. In YOLO, x and y determine the center of the object to be detected relative to the width and height of the entire image. And w and h define the width and height of the object to be detected, the same way.

An important part of the operation of YOLO is the use of anchor boxes that are used from version v2. These are made by the clustering of all w and h values from the training database. And from the B bounding box outputs, selection is done by the similarity to the anchor boxes. (Redmon and Farhadi, 2017)

2.2 The main new feature in YOLOv3

There are several newer versions of YOLO. A significant change in v3 is that there are more upscalings. So not only an $S \cdot S$ grid is used, but also a $2S \cdot 2S$ grid and a $3S \cdot 3S$ grid. This is very useful because, as mentioned in Introduction, YOLOv3 is designed to improve the detection of small objects. This can be done with the higher resolution activation map. (Redmon and Farhadi, 2018)

2.3 Effective new features in YOLOv4

The architecture of the v4 version is similar to the architecture of the v3, but it brings a lot of

small ideas and innovations. These new ideas can basically be divided into two groups. The first group is the Bag of Freebies. Methods that fall into this group do not result in additional computational costs, i.e., they do not increase runtime. So, for example Data Augmentation is used in v4 to create new and different training data. Another method of Bag of Freebies is to modify the Loss Function. This is necessary because a feature of YOLO architectures is that they generate far more bounding boxes than there are real objects to detect. Thus, false negative detections should be punished to a much greater extent than false positive detections. But there are many other features in this group as well (Bochkovskiy et al, 2020).

The second group is the Bag of Specials. With the methods belonging to this group, a significant improvement can be achieved in addition to a small increase in the computational costs. This includes, for example, the use of Mish Activation and Spatial Pyramid Pooling but there are other features as well (Bochkovskiy et al, 2020). Mish (Misra, 2019), which improves the performance of the neural network architecture, is defined using Softplus function, that is a smooth approximation to the ReLU activation function:

$$f(x) = x \cdot \tanh(\text{softplus}(x)) \quad (1)$$

It is a self-regularized non-monotonic activation function and works better than Leaky-ReLU that is the activation function of YOLOv3. In YOLOv4 features are pooled and fixed-length, outputs are generated by Spatial Pyramid Pooling (He et al, 2015), which is a layer added on top of the last convolutional layer. In the field of backbone usage YOLOv4 can use the cross-stage partial (CSP) network that can enhance the learning capability of the architecture. (Wang et al, 2020a)

2.4 The YOLOv5 architecture by Ultralytics

The name of this architecture entails a lot of controversy among those involved in object detection. The reason for this is, first of all, that YOLOv5 was released on May 27, 2020, by then-unknown authors, just over a month after YOLOv4 was released on April 23, 2020. Furthermore, it contained only minor changes compared to YOLOv4.

Moreover, as it turned out later, it is developed by a startup called Ultralytics, which was founded by YOLOv4’s mosaic data augmentation specialist, Glenn Jocher. That is why YOLOv5 does not have an original publication, only the Ultralytics GitHub repository (bib, 2020b). In this work the YOLOv5 name is used for the algorithm developed by Jocher et al.

A lot of paper has already compared v4 and v5. Ghimire et al (2022) highlights that YOLOv5 is much easier to integrate into robotic systems because it is implemented entirely in Pytorch. Li et al (2021) adds that although YOLOv5 uses CSPDarknet as a backbone, PANet as a neck, and the head of YOLOv3 as a head, but its activation function is a sigmoid function instead of Mish. Wang et al (2022) concludes that the usage of GIoU loss function effectively solves the problem of nonoverlapping bounding boxes. These are the main new features of YOLOv5. Apart from these, it is significantly similar to YOLOv4.

2.5 Data management

The used dataset is in a GitHub repository provided by Tao et al (2020). It is divided into two parts. The *NormalInsulators* part contains 600 Portable Network Graphics (PNG) pictures captured by UAVs. The PNG pictures about defective insulators are in the other part, that is called *DefectiveInsulators* part. There are 248 of them. Since there are not too many defective insulators, a data augmentation method is applied. All of these 248 images are synthesized by the following method (Tao et al, 2020).

1. The first step is the segmentation of some of the defective insulators in the original picture by TVSeg (Unger et al, 2008). This is how the first mask images are obtained.
2. The second step is the augmentation of the original images and their masks that results in a lot of original-mask image pairs.
3. The following stage is the training of the U-Net (Ronneberger et al, 2015) with these pairs.
4. Then, the trained U-Net can segment the rest of the defective insulators.
5. And finally, the segmented parts can be attached to different backgrounds.

Both the images of *NormalInsulators* and images of *DefectiveInsulators* are basically in two

images subdirectories and their annotations are in two labels subdirectories. Only the insulator annotations are used in this work.

The original annotation format is the Pascal Visual Object Classes (VOC) format, so it has to be converted to the input format of the YOLO architecture. It is done by the following equations:

$$x_{center}^{YOLO} = \frac{x_{min}^{VOC} + x_{max}^{VOC}}{2 \cdot w_{picture}^{VOC}} \quad (2)$$

$$y_{center}^{YOLO} = \frac{y_{min}^{VOC} + y_{max}^{VOC}}{2 \cdot h_{picture}^{VOC}} \quad (3)$$

$$w^{YOLO} = \frac{x_{max}^{VOC} - x_{min}^{VOC}}{w_{picture}^{VOC}} \quad (4)$$

$$h^{YOLO} = \frac{y_{max}^{VOC} - y_{min}^{VOC}}{h_{picture}^{VOC}} \quad (5)$$

where,

- x_{min}^{VOC} is the minimum horizontal pixel value of the insulator;
- x_{max}^{VOC} is the maximum horizontal pixel value of the insulator;
- y_{min}^{VOC} is the minimum vertical pixel value of the insulator;
- y_{max}^{VOC} is the maximum vertical pixel value of the insulator;
- $w_{picture}^{VOC}$ is the width of the entire picture in pixels;
- $h_{picture}^{VOC}$ is the height of the entire picture in pixels;
- x_{center}^{YOLO} is the horizontal center of the insulator relative to the entire picture;
- y_{center}^{YOLO} is the vertical center of the insulator relative to the entire picture;
- w^{YOLO} is the width of the insulator relative to the entire picture;
- and h^{YOLO} is the height of the insulator relative to the entire picture.

After these modifications, the training of the YOLO architectures can be performed.

3 Evaluation and Discussion of Results

This section presents the results of the architectures YOLOv5n (bib, 2020b), YOLOv5s (bib, 2020b), YOLOv5m (bib, 2020b), YOLOv5l (bib, 2020b), YOLOv5x (bib, 2020b), YOLOv5n6 (bib, 2020b), YOLOv5s6 (bib, 2020b), YOLOv5m6 (bib, 2020b), YOLOv5l6 (bib, 2020b), YOLOv4 (bib, 2020c), YOLOv4-tiny (bib, 2020c), YOLOv4-csp (bib, 2020c), YOLOv3 (bib, 2020a), YOLOv3-tiny (bib, 2020a) and YOLOv3-spp (bib, 2020a). In the YOLO loss-logic three components are considered with different weights. The first of them is the classification loss with weight 1, the second is the localization loss with weight 5 and the last is the confidence loss with weight 0.5. These are the default values in the YOLO architecture. They can of course be modified if required. The accuracy can be measured with mAP. The following subsection describes these methods. After that, results are presented.

3.1 Evaluation metrics

This section describes the calculation of the classification loss, the localization loss, the confidence loss and the mAP.

3.1.1 Classification loss

In the case of a detected object, the classification loss (Redmon et al, 2016) ($L_{classification}$) can be calculated with the sum of each squared error of the class conditional probabilities by classes and by cells:

$$L_{classification} = \sum_{i=0}^{S^2} I_i^{\text{object}} \cdot \sum_{\text{class} \in \text{classes}} (p_i(\text{class}) - \hat{p}_i(\text{class}))^2 \quad (6)$$

where,

- S^2 is the number of grid cells;
- $I_i^{\text{object}} = 0$ if an object does not appear in the i^{th} cell, otherwise 1;
- $\hat{p}_i(\text{class})$ is the predicted conditional class probability in the i^{th} cell;

- and $p_i(\text{class})$ is the conditional class probability in the i^{th} cell.

3.1.2 Localization loss

The localization loss (Redmon et al, 2016) ($L_{localization}$) composes of the fault of the boundary box size prediction and the fault of the boundary box location prediction:

$$L_{localization} = \sum_{i=0}^{S^2} \sum_{j=0}^B I_{i,j}^{\text{object}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (w_i^{\frac{1}{2}} - \hat{w}_i^{\frac{1}{2}})^2 + (h_i^{\frac{1}{2}} - \hat{h}_i^{\frac{1}{2}})^2] \quad (7)$$

where,

- S^2 is the number of grid cells;
- $I_i^{\text{object}} = 0$ if the j^{th} boundary box in the i^{th} cell is not responsible for the object detection, otherwise 1;
- x_i is the horizontal coordinate of the center of the boundary box;
- $\hat{x}_i$ is the prediction of the horizontal coordinate of the center of the boundary box;
- y_i is the vertical coordinate of the center of the boundary box;
- $\hat{y}_i$ is the prediction of the vertical coordinate of the center of the boundary box;
- w_i is the width of the boundary box;
- $\hat{w}_i$ is the predicted width of the boundary box;
- h_i is the height of the boundary box;
- and $\hat{h}_i$ is the predicted height of the boundary box.

The logic behind the exponentiation with $\frac{1}{2}$ is to consider the relative errors of the smaller boundary boxes similar to the relative errors of the large boundary boxes.

3.1.3 Confidence loss

The confidence loss (Redmon et al, 2016) ($L_{confidence}$), that measures the “objectness” of

the detected object in the boundary box is calculated as follows:

$$L_{\text{confidence}} = \sum_{i=0}^{S^2} \sum_{j=0}^B I_{i,j}^{\text{object}} \left(C_i - \hat{C}_i \right)^2 \quad (8)$$

where,

- S^2 is the number of grid cells;
- $I_i^{\text{object}} = 0$ if the j^{th} boundary box in the i^{th} cell is not responsible for the object detection, otherwise 1;
- C_i is the confidence score of the j^{th} boundary box in the i^{th} cell;
- and $\hat{C}_i$ is the prediction of the confidence score of the j^{th} boundary box in the i^{th} cell.

3.1.4 Mean Average Precision

Accuracy is measured in this paper with two metrics that use mAP (Padilla et al, 2020). The first one is the mAP_{50} that calculates the mAP at 50 % Intersection over Union (IoU) threshold by the following way:

$$mAP_{50} = \sum_{\text{class} \in \text{classes}} \int_0^1 p_{50}^{\text{class}}(r_{50}^{\text{class}}) \quad (9)$$

where,

- p_{50}^{class} is the precision of the class at 50 % IoU;
- and r_{50}^{class} is the recall of the class at 50 % IoU.

The other one is the $mAP_{50:95}$ that averages the mAP values at different IoU threshold levels from 50 % to 95 % with step size 5 %:

$$mAP_{50:95} = \frac{1}{10} \sum_{\text{perc}} \sum_{\text{class}} \int_0^1 p_{\text{perc}}^{\text{class}}(r_{\text{perc}}^{\text{class}}) \quad (10)$$

where,

- $p_{\text{perc}}^{\text{class}}$ is the precision of the class at perc % IoU;
- and $r_{\text{perc}}^{\text{class}}$ is the recall of the class at perc % IoU.

3.2 Results of different architectures

Results are highlighted in this subsection. All of them is obtained using an NVIDIA Quadro RTX

3000 GPU. Some more charts and visualizations about them can be found in the Appendix.

3.2.1 Results of the YOLOv3 architectures

Three subtypes of YOLOv3 architectures are evaluated in this paper in terms of insulator detection. These are the YOLOv3, the YOLOv3-tiny and the YOLOv3-spp (bib, 2020a).

The YOLOv3-tiny has much fewer layers than YOLOv3 and the difference between them in the number of parameters is even greater. This is because the simplified YOLOv3-tiny architecture, that requires less computing performance, can even run on a smartphone (Nakahara et al, 2020). The weakness of this architecture is usually the lower accuracy, as it is composed of 48 layers instead of 261 and consists of 8,669,002 parameters instead of 61,502,815.

In the case of YOLOv3-spp, the SPP stands for Spatial Pyramid Pooling. SPP is basically one of the Bag of Specials of YOLOv4 but can also be used for YOLOv3. It results in 269 layers and 62,551,903 parameters. The results of these architectures are shown in Figure 2.

It is noticeable, that the two bigger architectures (YOLOv3 and YOLOv3-spp) outperform the smaller one. The classification loss at the end of the 50 epochs is more or less the same in the three cases, but remarkable differences can be observed in the cases of localization loss and validation loss. Of these two metrics, which give a larger loss for YOLOv3-tiny, the second is particularly spectacular. This means that even the tiny architecture can recognize if a detected particular long rod insulator is defective or non-defective. However, the exact position of this insulator is not so well predicted by the simpler and much more uncertain solution. It is a general phenomenon that there is a greater difference between validation values than between training values. There is no significant difference between the traditional normal size method and the one using SPP, but the latter is somewhat more effective in terms of losses.

The latter difference is not observed for precision, recall, and mAPs, but the tiny method performs worse in these metrics as well. The mAP_{50} values achieved are 98.8 and 98.9 %, while the highest $mAP_{50:95}$ values are 90.5 % and 91.0 %

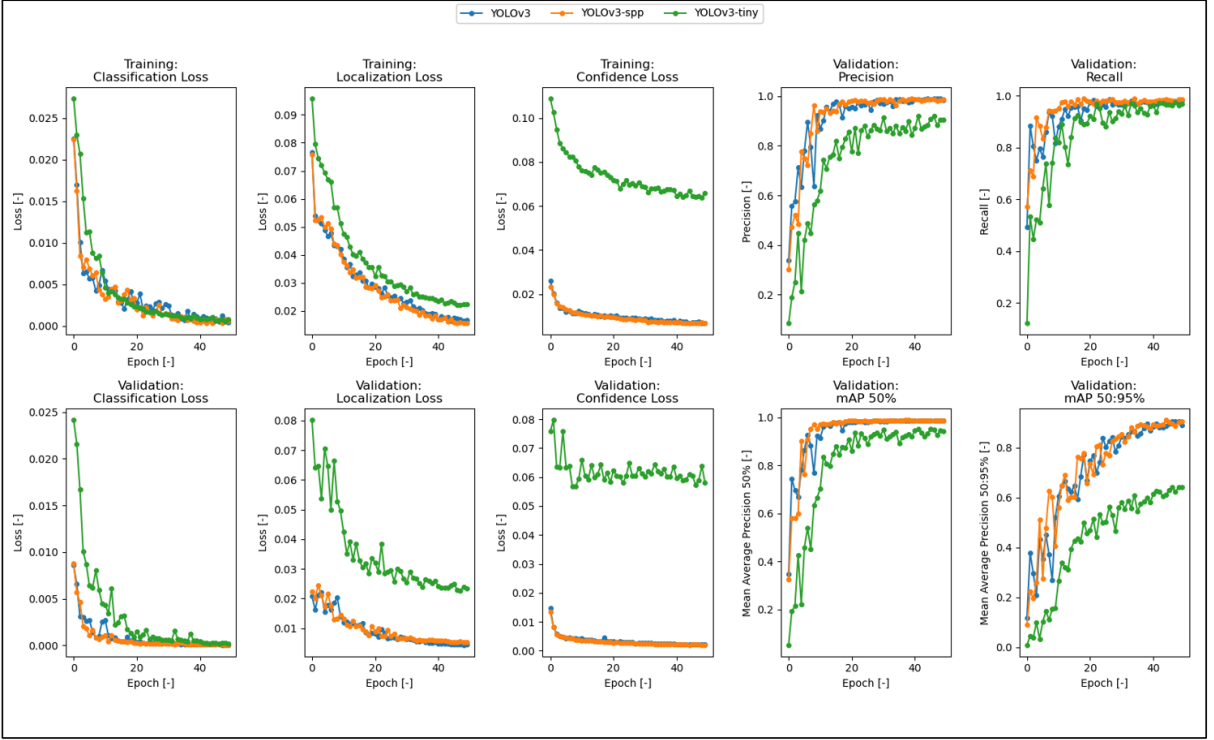


Fig. 2 Results of the YOLOv3 architectures

for YOLOv3 and YOLOv3-spp. These metrics indicate 95.1 % and 64.3 % for the tiny version. However, in terms of speed, this version performs better. The achievable FPS values are 56.8, 56.2, and 222.2, respectively.

3.2.2 Results of the YOLOv4 architectures

This paper summarizes the running results of three different YOLOv4 architectures. These are the YOLOv4, the YOLOv4-tiny and the YOLOv4-csp (bib, 2020c).

The difference between v4 and v4-tiny is similar to that between v3 and v3-tiny. The original YOLOv4 consists of 488 layers and 63,943,071 parameters. In contrast, YOLOv4 tiny consists of only 99 layers and 5,876,426 parameters. As a result, like YOLOv3-tiny, this architecture has a high FPS rate. The YOLOv4-csp version has a bit more layers because of the special backbone. But despite the 514 layers, it has only 52,501,407 parameters, which is less than the number of parameters in the original v4 architecture.

The results of these architectures are shown in Figure 3.

It is easy to see that the less accurate detections belong to the tiny version in this case as well. However, it is important to note that the difference is not as significant as for v3 versions. The classification loss and the localization loss during training are close to the values of the larger architectures. But in validation, the classification loss is different, and the difference in confidence loss is significant in both cases

The evaluations are very similar for precisions, recalls, and mAPs. This means that the best precision, recall, mAP_{50} and $mAP_{50:95}$ values for YOLOv4, YOLOv4-csp and YOLOv4-tiny are: 67.6 % - 63.0 % - 58.1 %, 98.8 % - 99.1 % - 99.1 %, 97.8 % - 97.4 % - 97.1 % and 79.1 % - 80.0 % - 79.1 %. This is very beneficial for YOLOv4-tiny, as it means that significant speed improvements can be achieved with little accuracy degradation. This is because the FPS values for the three architectures are 51.5, 76.3, and 270.2.

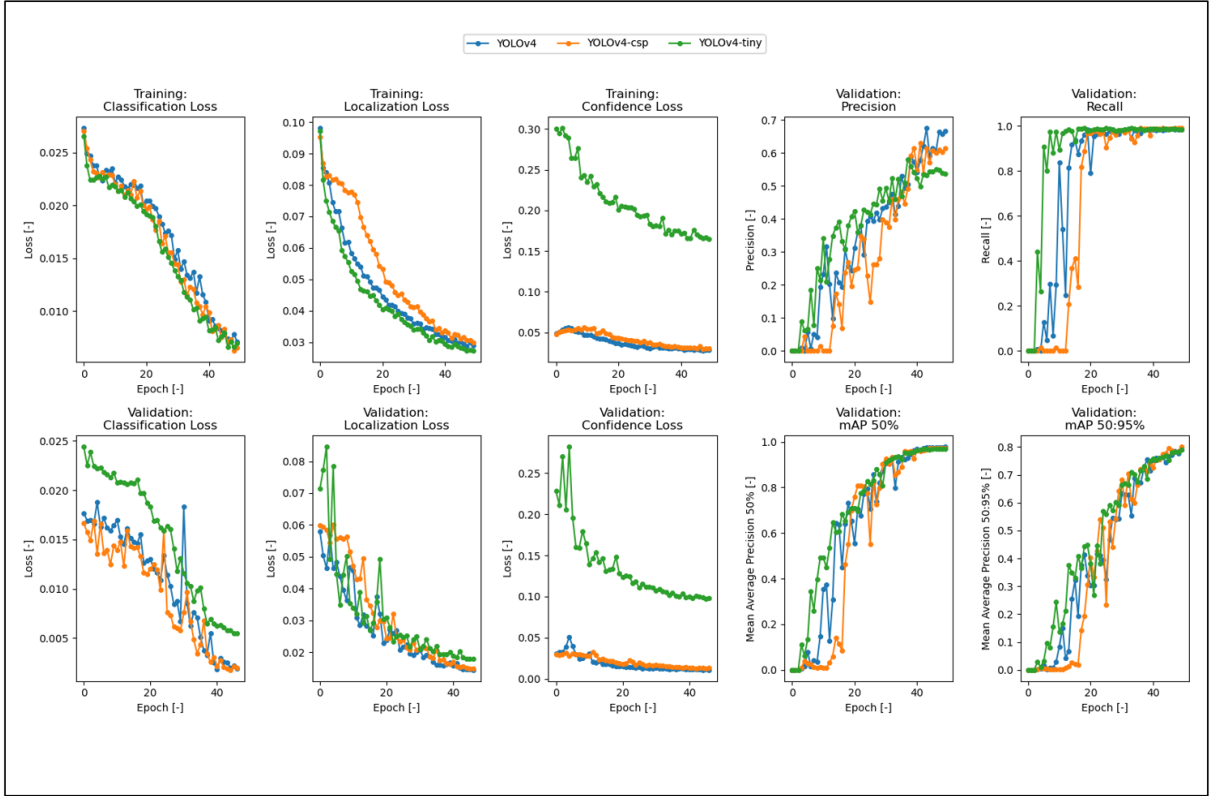


Fig. 3 Results of the YOLOv4 architectures

3.2.3 Results of the YOLOv5 architectures

The tested YOLOv5 algorithms are YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x, YOLOv5n6, YOLOv5s6, YOLOv5m6, and YOLOv5l6 (bib, 2020b). The tested YOLOv5 algorithms are YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x, YOLOv5n6, YOLOv5s6, YOLOv5m6, and YOLOv5l6 (bib, 2020b). The n, s, m, l and x letters are standing for the sizes: nano, small, medium, large and extra-large. Number 6 indicates preparation for higher resolution. Architectures marked with this number, include extra output layers for better results.

These architectures justify the “the bigger is the better is” principle. Training classification loss, validation classification loss, training localization loss, validation localization loss, training confidence loss, validation confidence loss, precision, recall, and mAPs are all the more favorable the more complex architectures are used. The best

loss values for v5 architectures are generally two times smaller than the best loss values for other architectures. Respectively, the decrease in losses is more monotonous than in the other cases.

The last four metrics vary from 96.9 % to 99.2 %, from 98.2 % to 99.4 %, from 98.1 % to 99.0 %, and from 83.0 % to 92.3 %. In these cases, size is the most significant influencing factor as well. However, as before, greater robustness means longer runtimes. Versions n, s, m, l, x, n6, s6, m6, and l6 have layers 213, 224, 316, 400, 484, 280, 280, 378, and 476, respectively. And their parameter numbers are 1,761,871, 7,056,607, 21,472,671, 47,370,047, 88,397,343, 3,089,188, 12,312,052, 35,254,692, and 76,126,356. These different solutions have FPS values of 212.8, 151.5, 84, 48.8, 27.1, 175.4, 137, 70.9, and 44.4. It is worth emphasizing that, for these methods, the correlation between the average image processing time ($AIPT = 1 / FPS$) and the number of layers is 0.917 and the parameter number - AIPT correlation is 0.966.

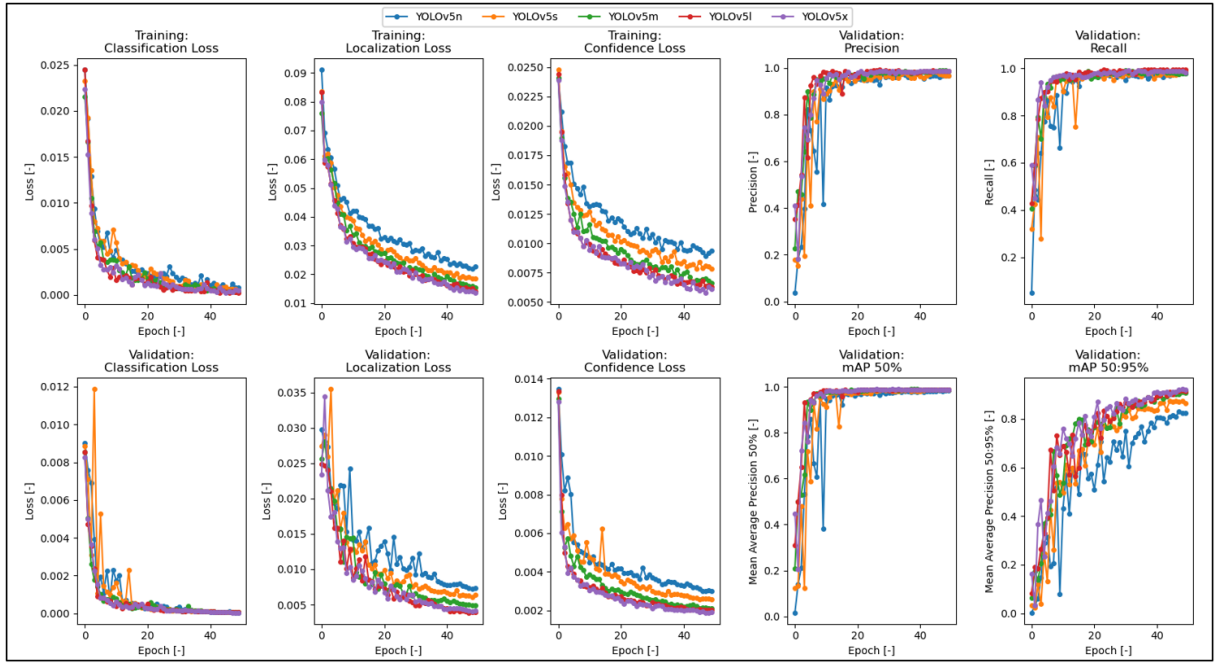


Fig. 4 Results of the YOLOv5 architectures

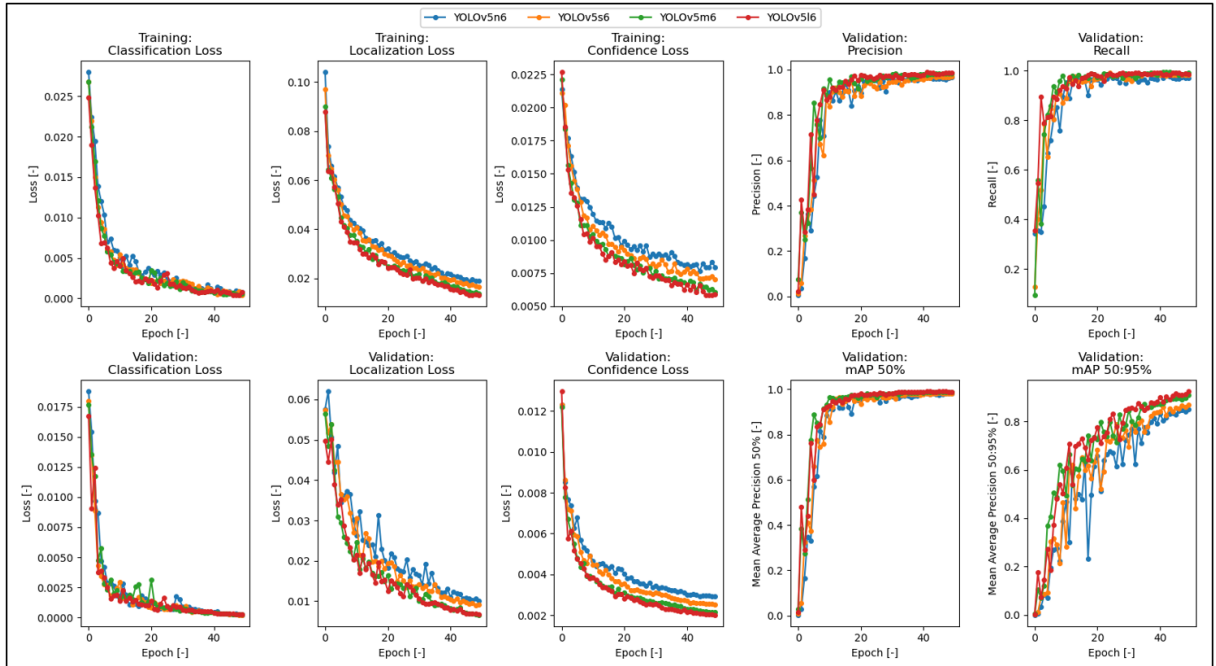


Fig. 5 Results of the YOLOv5 architectures with extra output layers

3.2.4 Same size architectures

Finally, it is worth comparing the subtypes of the different generations (v3, v4, v5) studied so far

in same size ranges. The v3-tiny, v4-tiny, v5n, v5s, v5n6, and v5s6 versions are similar. These architectures are small enough to reach 100 FPS.

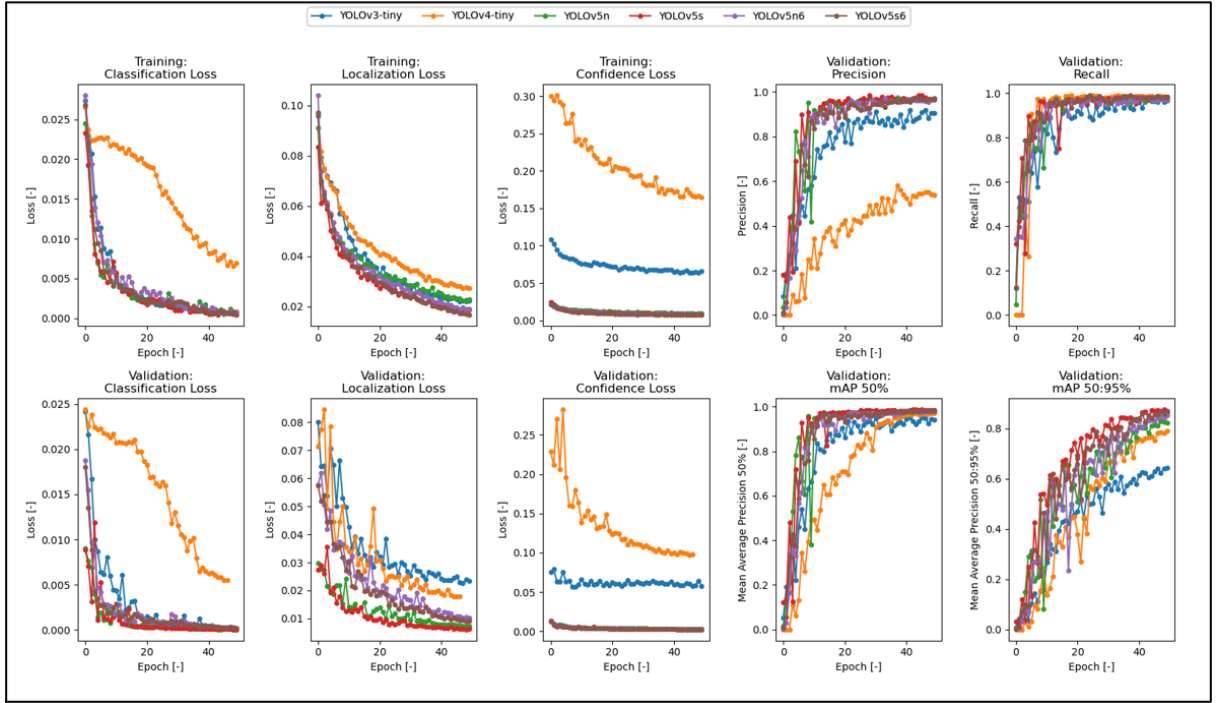


Fig. 6 Comparison of smaller architectures

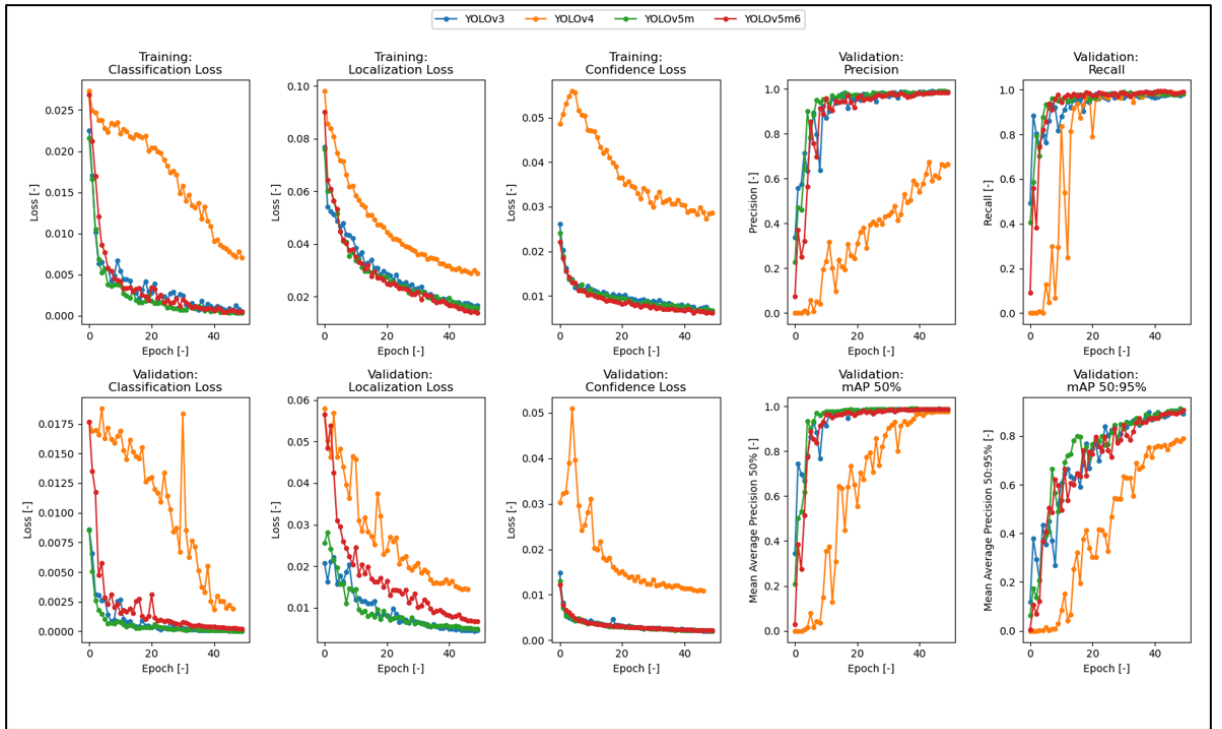


Fig. 7 Comparison of normal size architectures

Their comparison is shown in Figure 6. The same comparison is shown in Figure 7 for normal-sized original v3 and v4 and similar v5 architectures.

It is obvious that v5 architectures outperform v3 and v4. However, it should be noted that these have slightly lower FPS rates. The 5n, 5s, 5n6, and 5s6 architectures produced very similar results. That is why it may be advisable to choose the fastest of these architectures. The situation is similar for normal size. There, the m5 may be the best choice. An interesting phenomenon is that the v3 implementation of Ultralytics (bib, 2020a) is capable of better performance than the v4 implementation of Wang et al (2020a). However, typically the v4 architectures presented here can achieve higher FPS rates.

4 Summary

A benchmarking of YOLOv3, YOLOv4 and YOLOv5 was presented in this paper in terms of detection of defective and normal long rod insulators. Fifteen subtypes of YOLO architectures were evaluated on freely available dataset (Tao et al, 2020) based on the implementations (bib, 2020b), (bib, 2020c) and (bib, 2020a). The metrics selected for evaluation were training classification loss, validation classification loss, training localization loss, validation localization loss, training confidence loss, validation confidence loss, precision, recall, mAP_{50} , and $mAP_{50:95}$. FPS values were also considered during the evaluation of this work. It has been shown that, in general, subtypes of the YOLOv5 architecture are best suited for long rod insulator object detection. A complete evaluation of all architectures is shown in Table 1.

Future research will focus on the extension of the presented approach toward transfer learning and other scenarios.

Appendix A Extended Data

The following 15 figures show the results of each run separately.

Table 1 Best results of the different architectures

YOLO	L_{loc}^{train}	L_{loc}^{val}	L_{conf}^{train}	L_{conf}^{val}	L_{class}^{train}	L_{class}^{val}	Precision	Recall	mAP_{50}	$mAP_{50:95}$	FPS
v3-tiny	0.022055	0.022723	0.063935	0.05675	0.0005157	0.0001817	0.919	0.974	0.952	0.643	222.2
v3	0.0164	0.00442	0.00669	0.0021	0.000437	0.0000504	0.99	0.985	0.988	0.905	56.8
v3-spp	0.015511	0.0052551	0.0066339	0.0020084	0.0003049	0.0000563	0.99	0.988	0.989	0.91	56.2
v4-tiny	0.02719	0.01785	0.1649	0.09735	0.006578	0.005474	0.581	0.991	0.971	0.791	270.2
v4	0.02869	0.01445	0.02725	0.01085	0.007076	0.001848	0.676	0.988	0.978	0.791	51.5
v4-csp	0.02998	0.01492	0.02989	0.01281	0.006217	0.001795	0.63	0.991	0.974	0.8	76.3
v5n	0.021863	0.0072338	0.0089475	0.0029824	0.0005586	0.0000403	0.974	0.985	0.984	0.83	212.8
v5s	0.018647	0.00609	0.0078387	0.0025649	0.000472	0.0000508	0.984	0.985	0.987	0.875	151.5
v5m	0.015527	0.0048877	0.0066008	0.0020961	0.0003034	0.0000476	0.99	0.991	0.989	0.912	84
v5l	0.014116	0.003869	0.0062806	0.0020079	0.0001965	0.0000489	0.992	0.994	0.988	0.916	48.8
v5x	0.013462	0.0040794	0.0057686	0.0018832	0.0002368	0.0000431	0.987	0.991	0.989	0.922	27.1
v5n6	0.018701	0.010114	0.0076543	0.0029224	0.0006149	0.0002125	0.969	0.982	0.981	0.851	175.4
v5s6	0.016501	0.0090269	0.0070572	0.0025302	0.0004039	0.0002131	0.974	0.991	0.984	0.869	137
v5m6	0.01365	0.0066123	0.0060857	0.0021463	0.0004335	0.0002265	0.984	0.994	0.988	0.908	70.9
v5l6	0.013015	0.0067191	0.0058141	0.002025	0.0003554	0.0002435	0.989	0.991	0.99	0.923	44.4

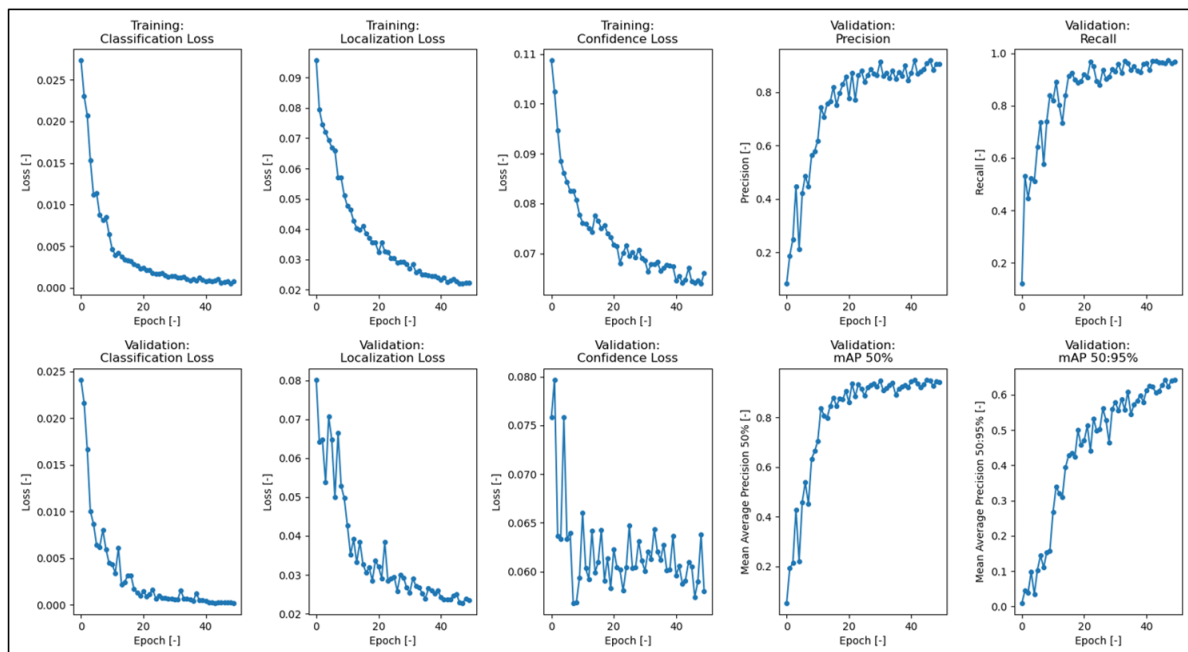


Fig. A1 Results of the YOLOv3-tiny architecture

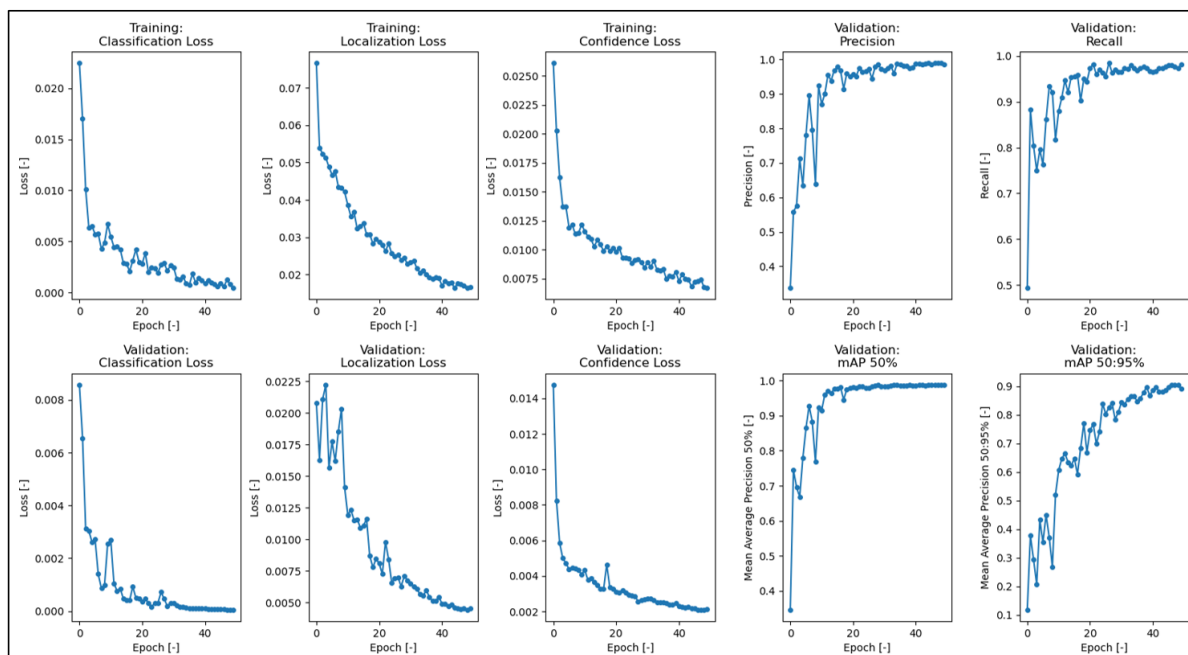


Fig. A2 Results of the YOLOv3 architecture

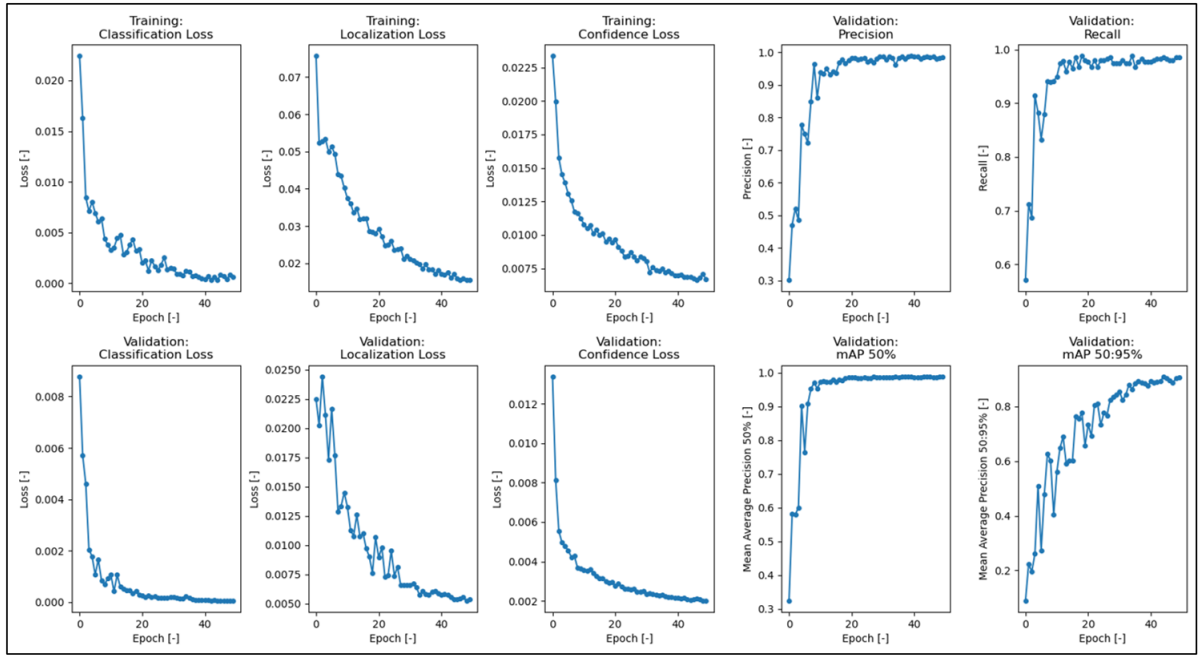


Fig. A3 Results of the YOLOv3-spp architecture

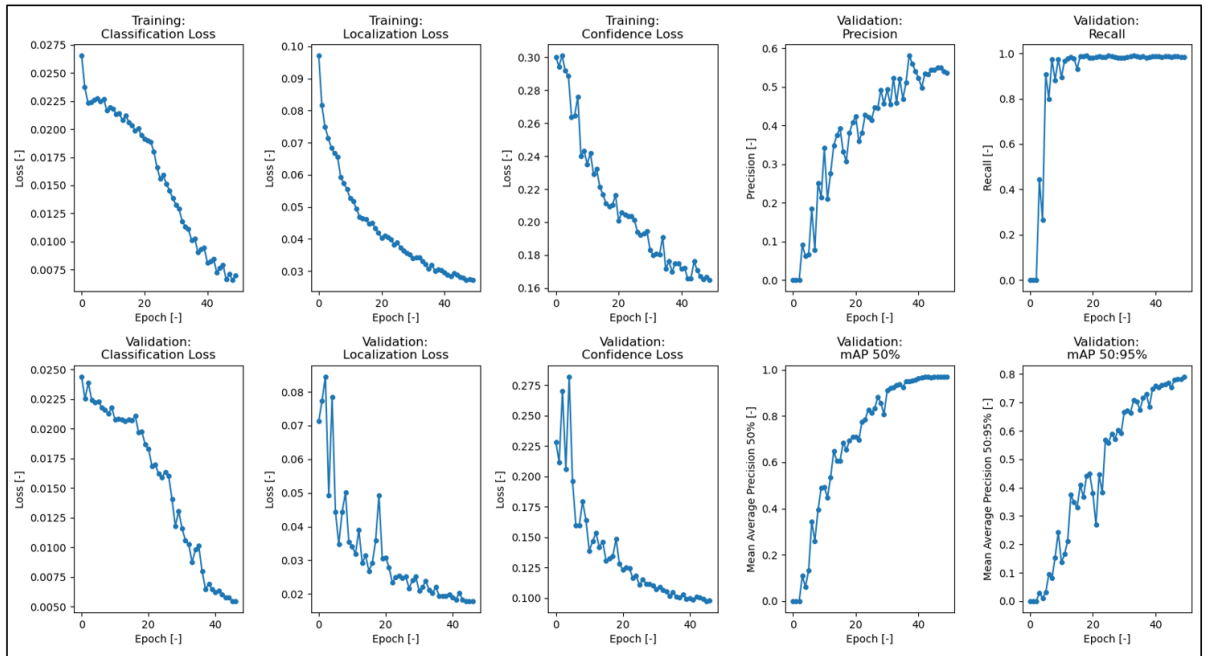


Fig. A4 Results of the YOLOv4-tiny architecture

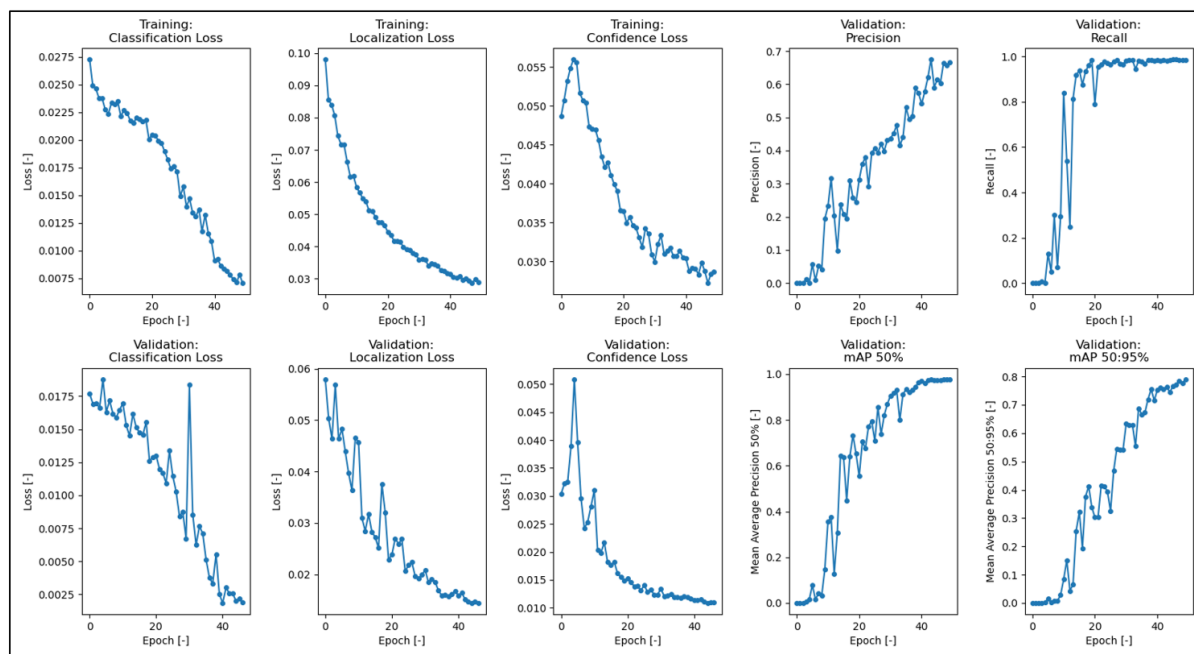


Fig. A5 Results of the YOLOv4 architecture

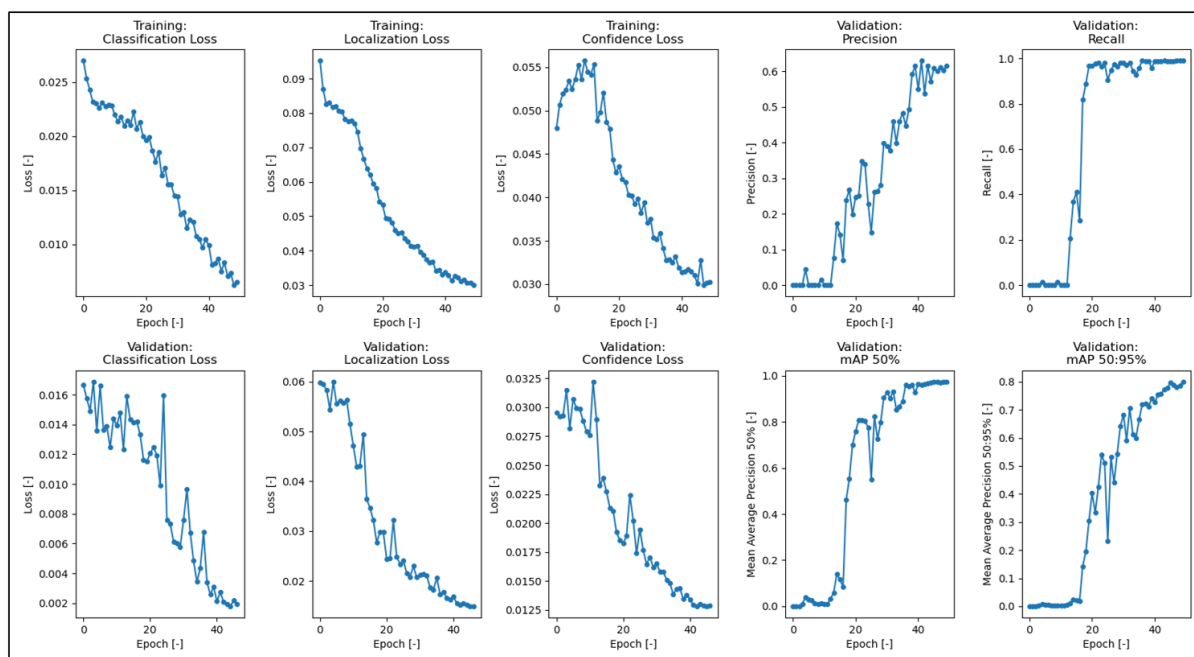


Fig. A6 Results of the YOLOv4-csp architecture

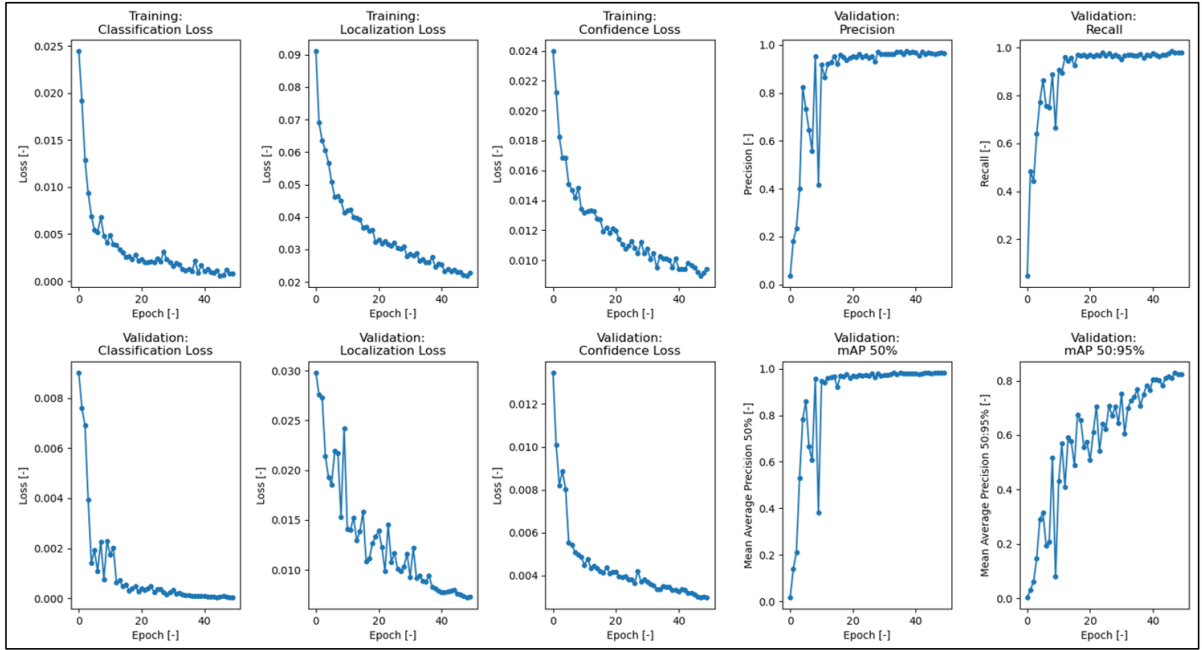


Fig. A7 Results of the YOLOv5n architecture

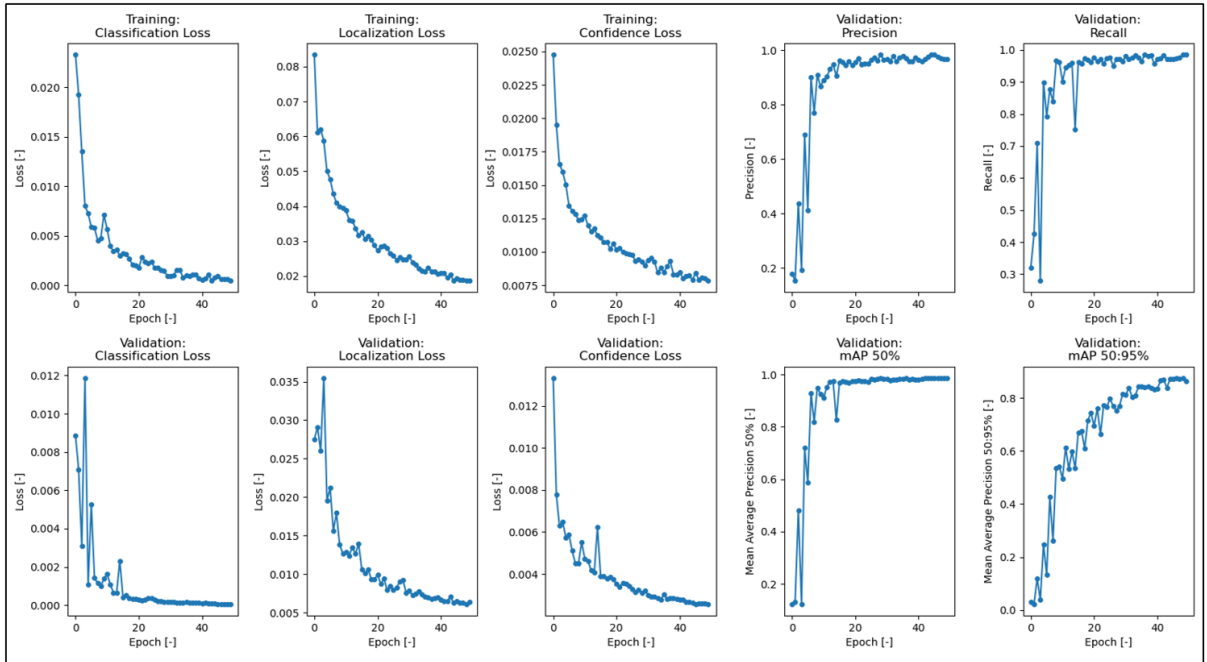


Fig. A8 Results of the YOLOv5s architecture

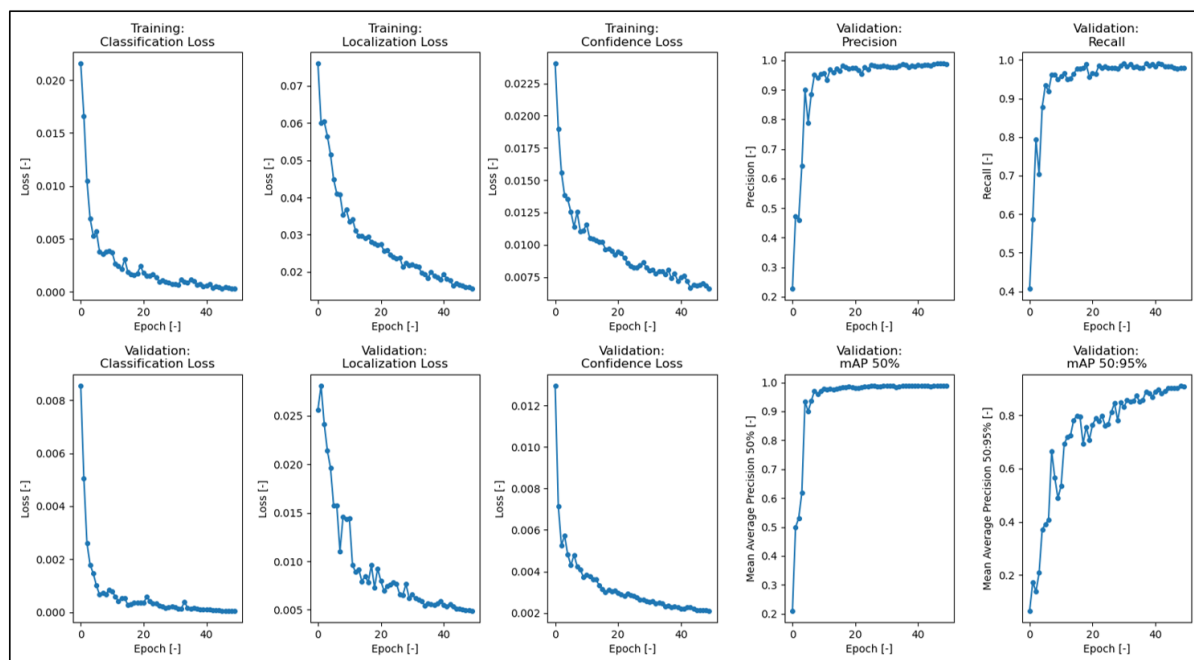


Fig. A9 Results of the YOLOv5m architecture

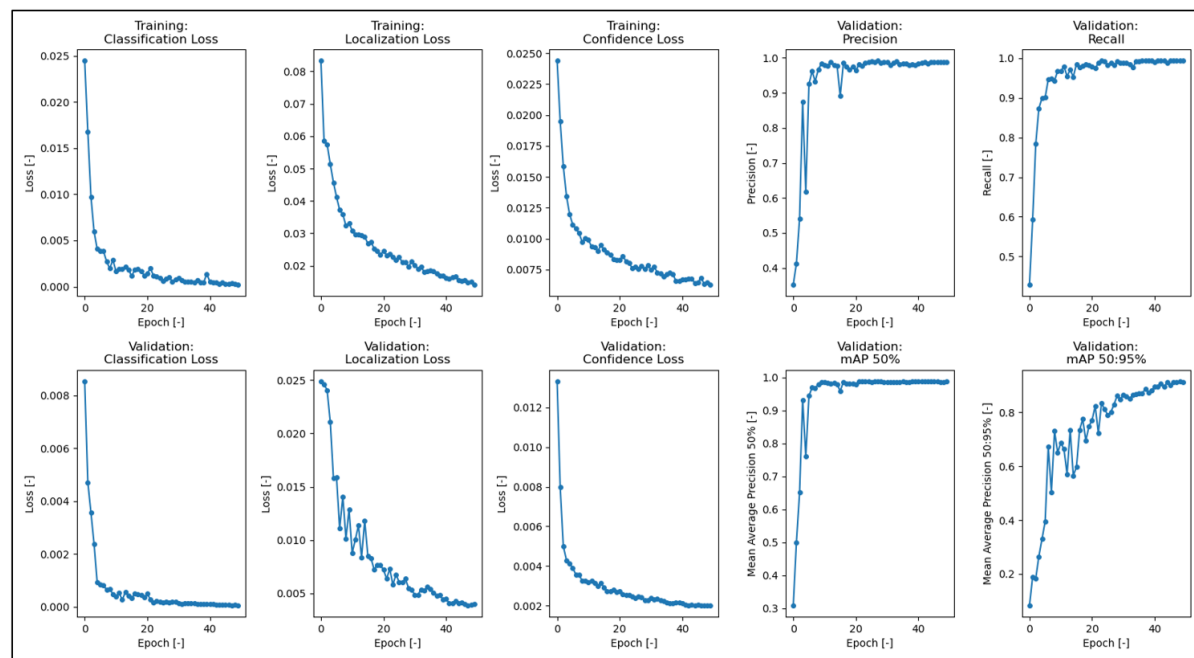


Fig. A10 Results of the YOLOv5l architecture

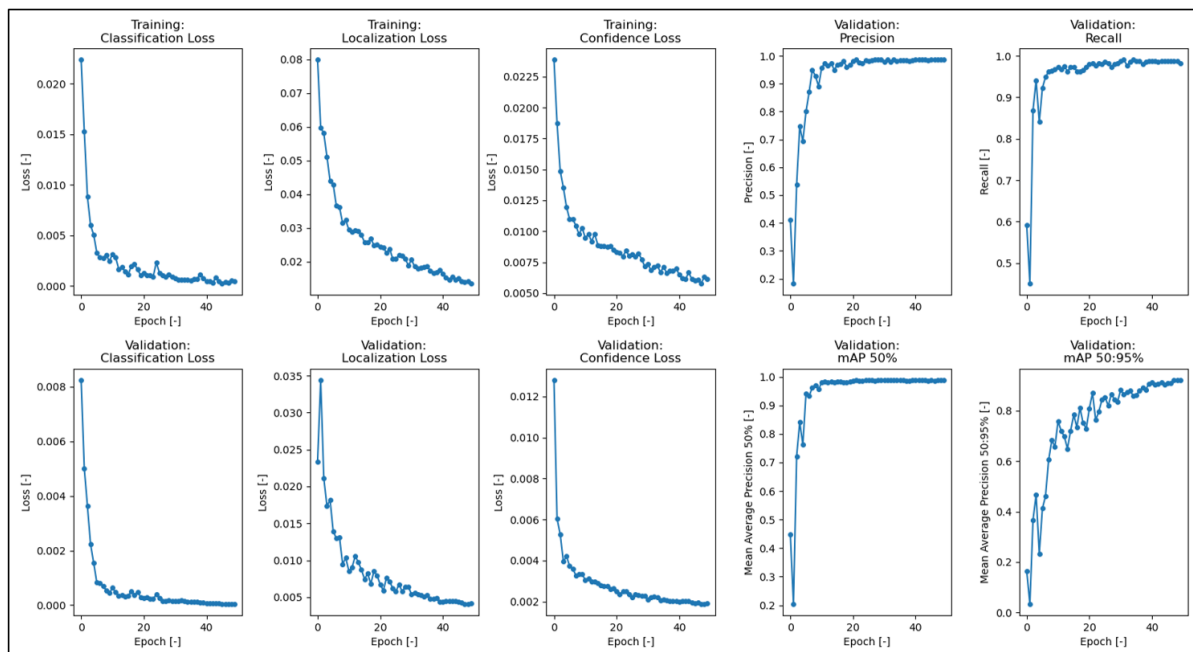


Fig. A11 Results of the YOLOv5x architecture

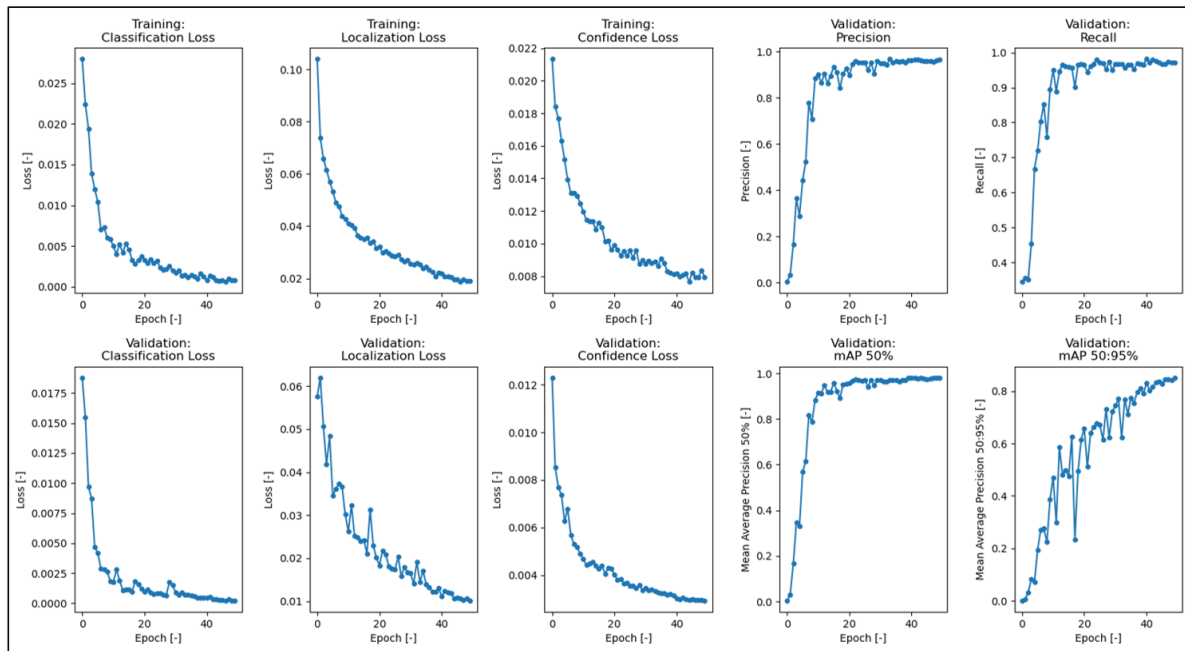


Fig. A12 Results of the YOLOv5n6 architecture

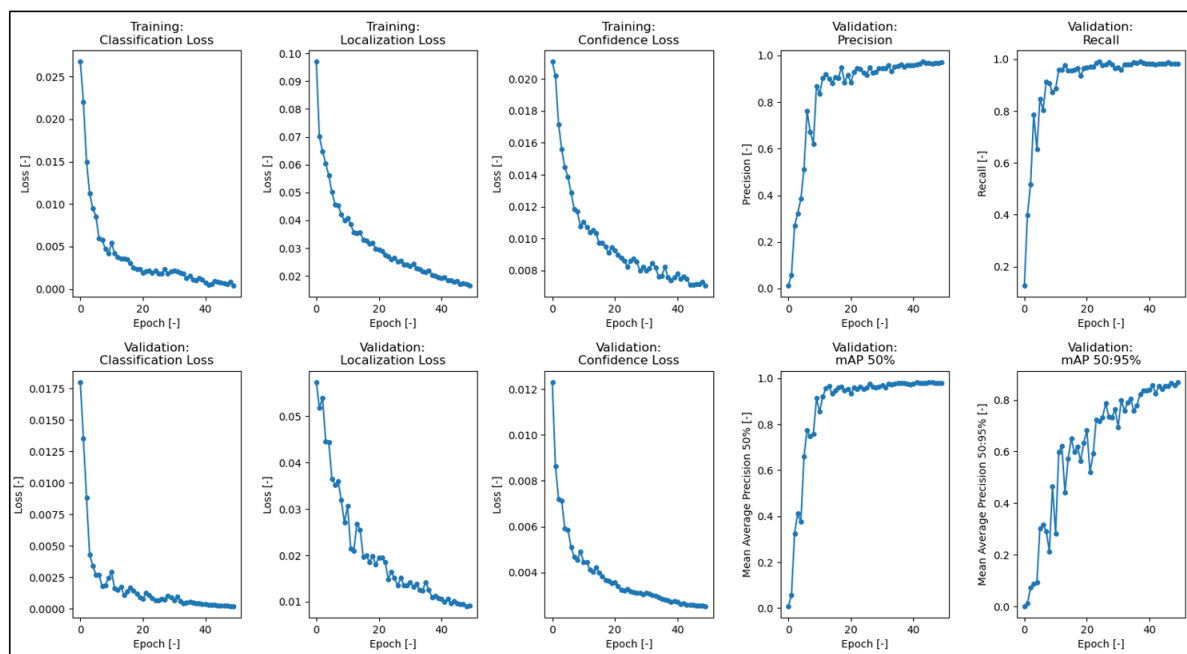


Fig. A13 Results of the YOLOv5s6 architecture

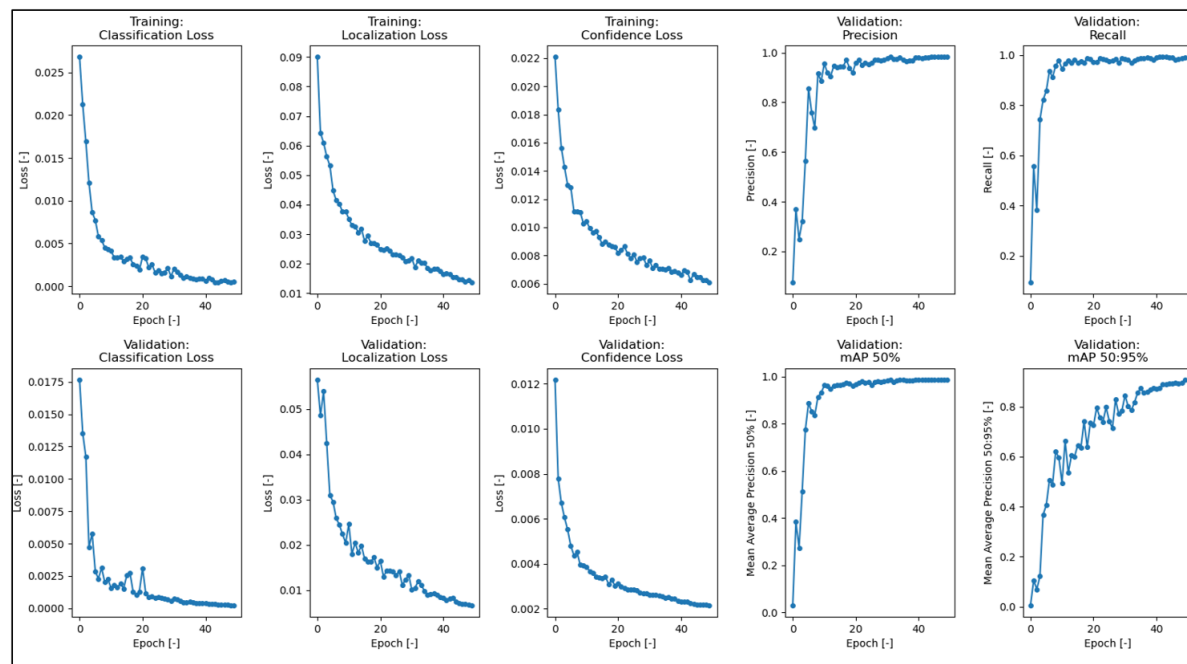


Fig. A14 Results of the YOLOv5m6 architecture

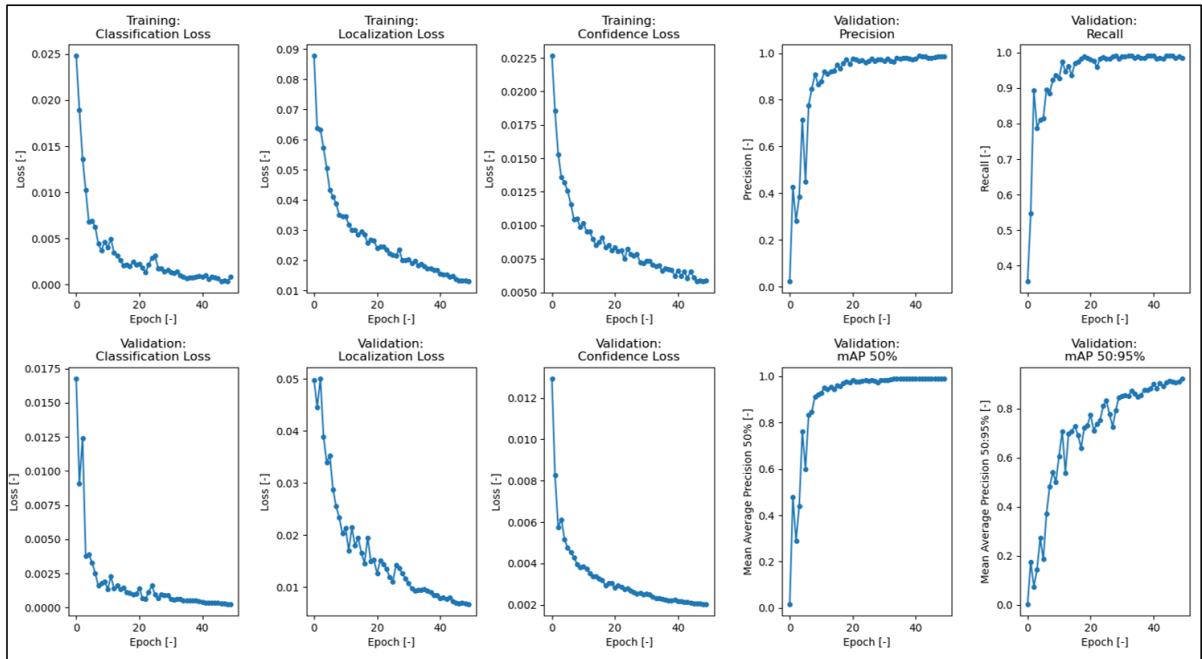


Fig. A15 Results of the YOLOv5l6 architecture

References

- (2020a) Ultralytics. yolov3. <https://github.com/ultralytics/yolov3>, accessed: 2022-07-26
- (2020b) Ultralytics. yolov5. <https://github.com/ultralytics/yolov5>, accessed: 2022-07-26
- (2020c) Wongkinyiu yolov4. https://github.com/WongKinYiu/PyTorch_YOLOv4, accessed: 2022-07-26
- Al-qaness MAA, Abbasi AA, Fan H, et al (2021) An improved yolo-based road traffic monitoring system. Computing 103. <https://doi.org/10.1007/s00607-020-00869-8>
- Ardhianto P, Subiakto R, Lin CY, et al (2022) A deep learning method for foot progression angle detection in plantar pressure images. Sensors 22(7). <https://doi.org/10.3390/s22072786>
- Avazov K, Mukhiddinov M, Makhmudov F, et al (2022) Fire detection method in smart city environments using a deep-learning-based approach. Electronics 11(1). <https://doi.org/10.3390/electronics11010073>
- Bochkovski A, Wang CY, Liao HY (2020) Yolo4: Optimal speed and accuracy of object detection. Preprint at <https://arxiv.org/pdf/2004.10934.pdf>
- Celik M (2022) Deep learning based detection tool for impacted mandibular third molar teeth. Diagnostics 12(4). <https://doi.org/10.3390/diagnostics12040942>
- Chen B, Miao X (2020) Distribution line pole detection and counting based on yolo using uav inspection line video. Journal of Electrical Engineering and Technology 15. <https://doi.org/10.1007/s42835-019-00230-w>
- Chen W, Huang H, Peng S, et al (2021) Yolo-face: a real-time face detector. The Visual Computer 37. <https://doi.org/10.1007/s00371-020-01831-7>
- Chen W, Li Y, Zhao Z (2022a) Missing-sheds granularity estimation of glass insulators using deep

- neural networks based on optical imaging. *Sensors* 22(5). <https://doi.org/10.3390/s22051737>
- Chen Z, Wu R, Lin Y, et al (2022b) Plant disease recognition model based on improved yolov5. *Agronomy* 12(2). <https://doi.org/10.3390/agronomy12020365>
- Cheng L, Li J, Duan P, et al (2021) A small attentional yolo model for landslide detection from satellite remote sensing images. *Landslides* 18. <https://doi.org/10.1007/s10346-021-01694-6>
- Dewi C, Chen R, Jiang X, et al (2022) Deep convolutional neural network for enhancing traffic sign recognition developed on yolo v4. *Multimedia Tools and Applications* <https://doi.org/10.1007/s11042-022-12962-5>
- Felzenszwalb P, McAllester D, Ramanan D (2008) A discriminatively trained, multiscale, deformable part model. In *Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, USA, 23–28 June 2008; pp. 1–8.
- Feng H, Jiang Y (2021) Recognition of insulator based on yolov5 algorithm. 2021 IEEE 11th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), Jiaxing, China, 27-31 July 2021.; IEEE: Jiaxing, China, 2021.; pp. 505-509
- Feng Z, Guo L, Huang D, et al (2021) Electrical insulator defects detection method based on yolov5. *IEEE 10th Data Driven Control and Learning Systems Conference (DDCLS)*, Suzhou, China, 14-16 May
- Fu HX, Song G, Wang YC (2021) Improved yolov4 marine target detection combined with cbam. *Symmetry* 13(4). <https://doi.org/10.3390/sym13040623>
- Gai R, Chen N, Yuan H (2021) A detection algorithm for cherry fruits based on the improved yolo-v4 model. *Neural Computing and Applications* <https://doi.org/10.1007/s00521-021-06029-z>
- Gao J, Chen X, Lin D (2021) Insulator defect detection based on improved yolov5. 2021 5th Asian Conference on Artificial Intelligence Technology (ACAIT), Haikou, China, 29-31 October
- Ghimire A, Werghi N, Javed S, et al (2022) Real-time face recognition system. Preprint at <https://arxiv.org/pdf/2204.08978.pdf>
- He K, Zhang X, Ren S, et al (2015) Spatial pyramid pooling in deep convolutional networks for visual recognition. Preprint at <https://arxiv.org/pdf/1406.4729.pdf>
- Hu H, Liu Y, Rong H (2022) Detection of insulators on power transmission line based on an improved faster region-convolutional neural network. *Algorithms* 15(3). <https://doi.org/10.3390/a15030083>
- Hu J, Shi CR, Zhang J (2021) Saliency-based yolo for single target detection. *Knowledge and Information Systems* 63. <https://doi.org/10.1007/s10115-020-01538-0>
- Junos MH, Mohd KAS, Thannirmalai S, et al (2022) Automatic detection of oil palm fruits from uav images using an improved yolo model. *The Visual Computer* 38. <https://doi.org/10.1007/s00371-021-02116-3>
- Karaci A (2022) Vggcov19-net: automatic detection of covid-19 cases from x-ray images using modified vgg19 cnn architecture and yolo algorithm. *Neural Computing and Applications* 34. <https://doi.org/10.1007/s00521-022-06918-x>
- Kim JA, Sung JY, Park SH (2020) Comparison of faster-rcnn, yolo, and ssd for real-time vehicle type recognition. In *Proceedings of the 2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, Seoul, Korea, 1–3 November 2020
- Kumar A, Kalia A, Sharma A, et al (2021) A hybrid tiny yolo v4-spp module based improved face mask detection vision system. *Journal of Ambient Intelligence and Humanized Computing* <https://doi.org/10.1007/s12652-021-03541-x>

- Lee Y, Kim Y (2020) Comparison of cnn and yolo for object detection. *Journal of the Semiconductor and Display Technology* 19:85–92
- Li B, Wu D, Cong Y, et al (2012) A method of insulator detection from video sequence. In *Proceedings of the 2012 International Symposium on Information Science and Engineering (ISISE)*, Shanghai, China, 14–16 December 2012; pp. 386–389.
- Li M, Zhang Z, Lei L, et al (2020) Agricultural greenhouses detection in high-resolution satellite images based on convolutional neural networks: Compar-ison of faster r-cnn, yolo v3 and ssd. *Sensors* 20(17)
- Li S, Gu X, Xu X, et al (2021) Detection of concealed cracks from ground penetrating radar images based on deep learning algorithm. *Construction and Building Materials* 273(10). <https://doi.org/10.1016/j.conbuildmat.2020.121949>
- Liu C, Wu Y, Liu J, et al (2021a) Improved yolov3 network for insulator detection in aerial images with diverse background interference. *Electronics* 10(7). <https://doi.org/10.3390/electronics10070771>
- Liu C, Wu Y, Liu J, et al (2021b) Insulator faults detection in aerial images from high-voltage transmission lines based on deep learning model. *Applied Sciences* 11(10). <https://doi.org/10.3390/app11104647>
- Liu J, Liu C, Wu Y, et al (2021c) An improved method based on deep learning for insulator fault detection in diverse aerial images. *Energies* 14(14). <https://doi.org/10.3390/en14144365>
- Liu M, Wang X, Zhou A, et al (2020) Uav-yolo: Small object detection on unmanned aerial vehicle perspective. *Sensors* 20(8). <https://doi.org/10.3390/s20082238>
- Loey M, Manogaran G, Taha M, et al (2021) Fighting against covid-19: A novel deep learning model based on yolo-v2 with resnet-50 for medical face mask detection. *Sustainable Cities and Society* 65:102,600. <https://doi.org/10.1016/j.scs.2020.102600>
- Ma H, Liu Y, Ren Y, et al (2020) Detection of collapsed buildings in post-earthquake remote sensing images based on the improved yolov3. *Remote Sensing* 12(1). <https://doi.org/10.3390/rs12010044>
- Ma L, Xu C, Zuo G (2017) Detection method of insulator based on faster r-cnn. In *Proceedings of the 2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, Honolulu, HI, USA, 31 July–4 August 2017; IEEE: Honolulu, HI, USA,
- Macey RE, Vosloo W, Tourreil C (2004) *The practical guide to outdoor high voltage insulators*. Crown Publications, Johannesburg
- Misra D (2019) Mish: A self regularized non-monotonic activation function. Preprint at <https://arxiv.org/pdf/1908.08681.pdf>
- Nakahara T, Fukuyama K, Hamada M, et al (2020) Mobile device-based speech enhancement system using lip-reading. In *Distributed Computing and Artificial Intelligence, 17th International Conference (DCAI)*, L'Aquila, Italy, 17–19 June, 2020; pp. 159–167.
- Padilla R, Netto S, Da Silva E (2020) A survey on performance metrics for object-detection algorithms. In *Proceedings of the 2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, Niteroi, Brazil, 1–3 July 2020; p. 6.
- Redmon J, Farhadi A (2017) Yolo9000: Better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
- Redmon J, Farhadi A (2018) Yolov3: An incremental improvement. Preprint at <https://arxiv.org/abs/1804.02767.pdf>
- Redmon J, Divvala S, Girshick R, et al (2016) You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.

- Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention; Springer: New York, NY, USA, 2015; pp. 234–241.
- Shao Y, Zhang X, Chu H, et al (2022) Air-yolov3: Aerial infrared pedestrian detection via an improved yolov3 with network pruning. *Applied Sciences* 12(7). <https://doi.org/10.3390/app12073627>
- Shi P, Jiang Q, Shi C, et al (2021) Oil well detection via large-scale and high-resolution remote sensing images based on improved yolo v4. *Remote Sensing* 13(16). <https://doi.org/10.3390/rs13163243>
- Sumit S, Watada J, Roy A, et al (2020) In object detection deep learning methods, yolo shows supremum to mask r-cnn. The 2nd Joint International Conference on Emerging Computing Technology and Sports (JICETS) 2019 25-27 November 2019, Bandung, Indonesia
- Tao X, Zhang D, Wang Z, et al (2020) Detection of power line insulator defects using aerial images analyzed with convolutional neural networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50(4). <https://doi.org/10.1109/TSMC.2018.2871750>
- Tomaszewski M, Michalski P, Osuchowski J (2020) Evaluation of power insulator detection efficiency with the use of limited training dataset. *Applied Sciences* 10(6). <https://doi.org/10.3390/app10062104>
- Unger M, Pock T, Trobin W, et al (2008) Tvseg-interactive total variation based image segmentation. In Proceedings of the British Machine Vision Conference (BMVC) 2008, England, UK, 1–4 September
- Van N, Nguyen R, Roverso D (2018) Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning. *International Journal of Electrical Power and Energy Systems* 99. <https://doi.org/10.1016/j.ijepes.2017.12.016>
- Viola P, Jones M (2001) Rapid object detection using a boosted cascade of simple features. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Kauai, HI, USA, 8–14 December 2001
- Wageeh Y, Mohamed HE, Fadl A, et al (2021) Yolo fish detection with euclidean tracking in fish farms. *Journal of Ambient Intelligence and Humanized Computing* 12. <https://doi.org/10.1007/s12652-020-02847-6>
- Wang CY, Liao HY, Wu YH, et al (2020a) Cspnet: A new backbone that can enhance learning capability of cnn. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Seattle, WA, USA, 14–19 June 2020; pp. 1571–1580.
- Wang J, Wang N, Li L, et al (2020b) Real-time behavior detection and judgment of egg breeders based on yolo v3. *Neural Computing and Applications* 32. <https://doi.org/10.1007/s00521-019-04645-4>
- Wang Y, Li Z, Yang X, et al (2020c) Insulator defect recognition based on faster r-cnn. In Proceedings of the 2020 International Conference on Computer, Information and Telecommunication Systems (CITS), Hangzhou, China, 5–7 October,
- Wang Z, Wu L, Li T, et al (2022) A smoke detection model based on improved yolov5. *Mathematics* 10(7). <https://doi.org/10.3390/math10071190>
- Xianbao C, Guihua Q, Yu J, et al (2021) An improved small object detection method based on yolo v3. *Pattern Analysis and Applications* 24. <https://doi.org/10.1007/s10044-021-00989-7>
- Xu Y, Zhang K, Wang L (2021) Metal surface defect detection using modified yolo. *Algorithms* 14(9). <https://doi.org/10.3390/a14090257>
- Yu J, Choi H (2022) Yolo mde: Object detection with monocular depth estimation.

Electronics 11(76). <https://doi.org/10.3390/electronics11010076>

Zhang X, An J, Chen F (2010) A method of insulator fault detection from airborne images. In Proceedings of the 2nd WRI Global Congress on Intelligent Systems, Wuhan, China, 16–17 December 2010; pp. 200–203.

Zhang Z, Xia S, Cai Y, et al (2021) A soft-yolov4 for high-performance head detection and counting. Mathematics 9(23). <https://doi.org/10.3390/math9233096>

Zhang Z, Huang S, Li Y, et al (2022) Image detection of insulator defects based on morphological processing and deep learning. Energies 15(7). <https://doi.org/10.3390/en1507246>

Zhao J, Li C, Xu Z, et al (2022) Detection of passenger flow on and off buses based on video images and yolo algorithm. Multimedia Tools and Applications 81. <https://doi.org/10.1007/s11042-021-10747-w>

Zhao Z, Zhen Z, Zhang L, et al (2019) Insulator detection method in inspection image based on improved faster r-cnn. Energies 12(7). <https://doi.org/10.3390/en12071204>

Zhou S, Bi Y, Wei X, et al (2021) Automated detection and classification of spilled loads on freeways based on improved yolo network. Machine Vision and Applications 32. <https://doi.org/10.1007/s00138-021-01171-z>

Zou X (2019) A review of object detection techniques. In Proceedings of the 2019 International Conference on Smart Grid and Electrical Automation (ICSGEA), Xiangtan, China, 10–11 August 2019; pp. 251–254.

Zou Z, Shi Z, Guo Y, et al (2019) Object detection in 20 years: A survey. Preprint at <https://arxiv.org/pdf/1905.05055.pdf>

Conflict of Interest Statement

I have no conflicts of interest to declare. I certify that the submission is original work and is not under review at any other publication.