A comparative study of YOLOv5 models performance for image localization and classification

Marko Horvat, Ljudevit Jelečević, Gordan Gledec
Faculty of Electrical Engineering and Computing, Department of Applied Computing
University of Zagreb
Unska 3, HR-10000 Zagreb, Croatia
{Marko.Horvat3, Ljudevit.Jelecevic, Gordan.Gledec}@fer.hr

Abstract. *YOLOv5 is one of the latest and often used versions of a very popular deep learning neural network used for various machine learning tasks, mainly in computer vision. The YOLO algorithm has steadily gained acceptance in the data science community due to its superior performance in complex and noisy data environments, availability, and ease of use in combination with widely used programming languages such as Python. This paper aims to compare different versions of the YOLOv5 model using an everyday image dataset and to provide researchers with precise suggestions for selecting the optimal model for a given problem type. The obtained results and the implemented YOLOv5 models are available for non-commercial use at: https://github.com/mhorvat/YOLOv5-models-comparison*

Keywords. computer vision, image classification, deep learning, deep convolutional neural networks, YOLO

1 Introduction

The rapid development of Deep Learning (DL) has accelerated the progress of related methods, algorithms and procedures in the fields of image processing and computer vision, offering a wide range of applications. Nowadays, artificial neural networks are now the standard tool for most computer vision tasks. Since 2012, with the invention of AlexNet, deep neural networks have been used in thousands of remote sensing applications (Krizhevsky, Sutskever & Hinton, 2012). Among others, methods such as R-CNNs (Region Based Convolutional Neural Networks) (Girshick et al., 2014) and Fast R-CNNs (Fast Region Based Convolutional Neural Networks) (Girshick, 2015) – representing a family of machine learning models developed for computer vision and specifically object detection, as well as, Spatial Pyramid Pooling-Nets (He et al., 2015), and "You Only Look Once" (YOLO) (Redmon et al., 2016) networks provide promising results on a variety of computer vision problems.

The YOLO is currently one of the most popular DL open-source frameworks for a wide range of machine learning (ML) tasks (Jiang et al., 2022). Additionally, YOLOv4 was the best real-time object detection algorithm in 2021 based on the MAP benchmark on the standard MS COCO dataset and the fastest real-time object detection algorithm (Lin et al., 2014; Bochkovskiy, Wang & Liao, 2020). The YOLO classifiers family is principally used in computer vision for object detection and image classification, for example (Krišto, Ivasic-Kos & Pobar, 2020; Du et al., 2021), in both off-line and real-time (Redmon & Farhadi, 2018a; Fang, Wang & Ren, 2019) settings. Apart from YOLO, other frequently used algorithms for object detection are the already mentioned R-CNN and Fast R-CNN.

The YOLO algorithm is a single-stage target recognition DL algorithm. It was first proposed in 2016 by (Redmon et al., 2016), which implements object recognition classification and localization with a single neural network. It has been widely used in object recognition ever since. YOLO has undergone development from v1 to v7 (Redmon & Farhadi, 2017; Redmon & Farhadi, 2018b; Bochkovskiy, Wang & Liao, 2020; Jiang et al., 2022). Currently, YOLOv6/v7 and YOLOv5, launched in 2022 and 2020, respectively, are the latest algorithm with many practical advantages over the previous versions (Solawetz, 2020; Nelson & Solawetz, 2020; Jiang et al., 2022). Some of the most important benefits of YOLOv5 compared to previous versions are smaller volume, higher speed, higher precision, and implementation in the ecologically mature PyTorch open-source ML framework. R-CNN belongs to the two-stage object recognition algorithms, which can effectively improve the problem of the target under test in the image, but the two-stage model is more complex and computationally intensive than the single-stage model. The YOLOv5 algorithm is commonly selected for data science projects to classify free datasets available on online repositories such as Kaggle. The

original algorithm is adapted by researchers or employed directly.

The rest of the paper is organized as follows; following the introduction, section 2 explains the YOLOv5 model. The features of the architecture and the main differences between YOLOv5 and its predecessors, in particular v3 and v4, are presented. The main differences between YOLO v5 and the latest versions v6/v7 are explained in Section 2.1. In Section 3, the experiment to comparatively study the performance of YOLOv5 models is described. The standardized experimental data set is also described in this section. The results of the conducted experiment are presented and discussed in the fourth section. The fourth section also compares the YOLOv5 models using quantitative indicators. The last section concludes the paper and provides suggestions for engineers and researchers to use the YOLOv5 model for various problems DL.

2 The YOLOv5 model

"You Only Look Once" is an object detection approach that processes input images in its entirety, unlike the older "sliding window" approach where specific image segments are individually classified often in multiple passes. This approach allows for faster processing because it does not have to perform many independent evaluations and it also improves accuracy since the global image context is made available to the whole neural network (Redmon & Farhadi, 2018a; Yang et al., 2020; Jocher, 2020a; Jocher, 2020b).

The YOLOv5 (Solawetz, 2020) is based on the YOLO detection architecture and uses several algorithm optimization strategies from the field of convolutional neural networks such as auto learning bounding box anchors, mosaic data augmentation, and the cross-stage partial network. Compared to earlier solutions the YOLO model was the first object detector to connect the procedure of predicting bounding boxes with class labels in an end-to-end differentiable network (Nelson & Solawetz, 2020).

The YOLOv5 network consists of three main components: 1) backbone, 2) neck, and 3) output. First, the input terminal performs data preprocessing tasks including mosaic data augmentation (Solawetz, 2020) and adaptive image filling. To be able to adapt to different datasets, YOLOv5 integrates adaptive anchor frame calculation on the input, so that it can automatically set the initial anchor frame size when the dataset changes.

The backbone is a convolutional neural network that aggregates and forms image features at different granularities. It mainly uses a cross-stage partial network (CSP) (Kim et al., 2019) and spatial pyramid pooling (SPP) (He et al., 2015) to extract feature maps of different sizes from the input image by multiple convolution and pooling. The BottleneckCSP architecture is used to reduce the amount of calculation and increase the speed of inference, while the SPP structure realizes the feature extraction from different scales for the same feature map, and can generate three-scale feature maps, which helps improve the detection accuracy. The neck neural network represents a series of layers to mix and combine image features and to pass them forward to the prediction. In the neck network, the feature pyramid structures of FPN and PAN are used. The FPN (Liu et al., 2016) structure conveys strong semantic features from the top feature maps into the lower feature maps. At the same time, the PAN (Wang et al., 2019) structure conveys strong localization features from lower feature maps into higher feature maps. These two structures jointly strengthen the feature extracted from different network layers in Backbone fusion, which further improves the detection capability. As a final detection step, the head output is mainly used to predict targets of different sizes on feature maps.

The YOLOv5 model includes 10 individual architectures named YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x, YOLOv5n6, YOLOv5s6, YOLOv5m6, YOLOv5l6, and YOLOv5x6 + TTA (Jocher, 2020a). However, commonly only the first five are considered for research: *nano*, *small*, *medium*, *large*, *xlarge*. The main difference between them lies in the number of feature extraction modules and convolution kernels, and subsequently – which is important from the practical perspective in working with the YOLO models – in the number of neural network parameters. Accordingly, the size of parameters' set for the YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x is 4 MB, 14 MB, 41 MB, 89 MB, and as much as 166 MB, respectively (Yang et al., 2020; Jocher, 2020b). In a previous study these five YOLOv5 models were compared using COCO val2017 corpus (Lin et al., 2014; Nelson & Solawetz, 2020). In this image classification experiment researchers employed NVIDIA V100 Tensor Core GPU and the dataset consisted of 5000 pictures separated in 80 classes. The results of this study are shown in Table 1.

For evaluation of object detection, a common way to determine if one object proposal was correct is *Intersection over Union* (IoU, IU). This method takes the set $\boldsymbol{A}$ of proposed object pixels and the set of true object pixels $\boldsymbol{B}$ and calculates:

$$IoU(\boldsymbol{A},\boldsymbol{B}) = \frac{\boldsymbol{A} \cap \boldsymbol{B}}{\boldsymbol{A} \cup \boldsymbol{B}};\ IoU(\boldsymbol{A},\boldsymbol{B}) \in [0,1] \qquad (1)$$

Commonly, the threshold $IoU > 0.5$ means that object detection was a hit; otherwise, if $IoU \leq 0.5$ it was a fail. For each object class c one can calculate the *average precision* (AP) as:

$$AP(c) = \frac{TP(c)}{TP(c) + FP(c)} \qquad (2)$$

where $TP(c)$ represents the number of true positive instances and $FP(c)$ the number of false positive instances for class c. For an arbitrary class c value $AP(c) = 1$ would represent a perfect detection and $AP(c) = 0$ the worst. Therefore, the evaluation metric *mean average precision* (mAP) over the set of all objects ($\boldsymbol{O}$) in a dataset can be expressed as:

$$mAP = \frac{1}{|\boldsymbol{O}|} \sum_{c \in \boldsymbol{O}} AP(c) \quad (3)$$

Thus, metric $mAP_{0.5:0.95}$ indicates mAP over different IoU thresholds, from 0.5 to 0.95, in 0.05 increments. Similarly, $mAP_{0.5}$ describes mAP for $IoU > 0.5$. In popular object detection challenges (i.e. COCO Object Detection Challenge) additional common performance metrics is used such as $mAP_{\text{IoU=.50:.05:.95}}$, $mAP_{\text{IoU=.50}}$ and $mAP_{\text{IoU=.75}}$ (Padilla et al., 2020).

Table 1. Comparison of YOLOv5 models on COCOval 2017 dataset. Adapted from (Nelson & Solawetz, 2020).

Model	$mAP_{0.5}$	$mAP_{0.5:0.95}$	Training duration [s]
YOLOv5n	46.0	28.4	6.3
YOLOv5s	56.0	37.2	6.4
YOLOv5m	63.9	45.2	8.2
YOLOv5l	67.2	48.8	10.1
YOLOv5x	68.9	50.7	12.1

The key benefits of the YOLOv5 compared to the previous versions are 1) simple installation requiring only the PyTorch and some lightweight python libraries, 2) fast training and reduction of experimentation costs, 3) functional inference ports on individual images, batch images, video feeds, or webcam ports, 4) intuitive layout based on standard file folder layout that is intuitive and easy to navigate while developing, and 5) easy translation to mobile devices such as smartphones and tablets with Core ML compatibility. Also, from the developers' perspective, it is essential to point out that YOLOv5 is available in the PyTorch framework using Jupyter notebooks or Google Colab (Google Colaboratory) tools.

2.1 The YOLO v6 and v7 models

The latest YOLO versions MT-YOLOv6 and YOLOv7 were published online in June and July 2022 ("YOLOv6", 2022; Wang, Bochkovskiy, Liao, 2022) after this paper was accepted for review. As reported, both versions outperform YOLOv5 in object detection accuracy, training and classification speed. However, it should be noted that v6 and v7 are not part of the official YOLO series. They were developed independently by different developer groups, so YOLOv5 is still the last official YOLO release.

According to information currently in the public domain, MT-YOLOv6 (developed by the Meituan company, hence the prefix "MT") is a single-stage object detection framework dedicated to industrial applications with a hardware-friendly, efficient design and high performance. YOLOv6 has fewer model types (lacking m/l/x). Model MT-YOLOv6-s achieved 43.1 mAP on the COCO val2017 dataset, compared to 37.4 mAP for YOLOv5-s, with similar inference speed. MT-YOLOv6 shows significant improvements with small object detection in densely packed scenarios ("YOLOv6" 2022).

Even less is known about YOLOv7, except that it is said to outperform all known object detectors in terms of speed and accuracy in the range of 5-160 FPS and has the highest accuracy 56.8% AP among all known real-time object detectors with 30 FPS or higher on GPU V100 (Wang, Bochkovskiy, Liao, 2022). Of course, to thoroughly compare the latest YOLO models, further experiments with different datasets and in different environments are needed (Wang, Bochkovskiy, Liao, 2022).

3 Image classification experiment

In the experiment reported in this paper, the YOLOv5 models' performance was compared on the Face Mask Detection image dataset (Maranhão, 2020). This dataset was created for medical face mask detection, and its properties are sufficient to compare YOLOv5 models. The presented study is preliminary, and additional datasets and performance measures that are standardized in object recognition challenges, such as the COCO Object Detection Challenge, should be used for a more comprehensive analysis.

The described experiment is conceptually similar to research (Yang et al., 2020; Sharma, 2020; Ieamsaard, Charoensook & Yammen, 2021; Jiang et al., 2021) where YOLOv3 and YOLOv5 models were trained on datasets designed to train deep neural networks to detect the human face and classify whether the person in the image is wearing a mask and wearing it properly. In (Yang et al., 2020) a system was constructed to detect a human face in a picture and classify whether the person is wearing a mask. The authors considered several different deep neural networks, Faster R-CNN, R-FCN, SDD, and YOLOv5s, and compared them. When training the YOLOv5 neural network, they used the AIZOOTech dataset (Chiang, 2020) and got slightly better results when training lasted more than 300 epochs. In the second paper, the authors trained YOLOv5 models on a face detection corpus and performed classification to identify images of a person wearing a medical mask. The dataset was created by downloading images from the Internet and manually tagging them. The classification performance was slightly worse than in the first paper. Possible reasons were training on only 100 epochs, and the training data set contained a small number of examples.

3.1 Experimental dataset

The Face Mask Detection dataset used for training is available at the Kaggle repository (Maranhão, 2020). The Face Mask Detection dataset is in Public Domain (CC0: Public Domain) and it is freely available for research, education and even commercial purposes. A sample of images in the Face Mask Detection dataset is shown in Fig. 1. The Face Mask Detection dataset images are separated in 3 labelled classes: 1) persons with mask ("with_mask"); 2) persons without mask ("without_mask"); 3) persons with mask worn incorrectly ("mask_weared_incorrect").

Figure 1. Example of images in the Face Mask Detection dataset used for the comparative study.

The image bounding boxes are available in the PASCAL VOC format. The original Face Mask Detection dataset contains 853 images. However, this was insufficient for training YOLOv5 models in this research. Therefore, all images in the dataset were mirrored along their horizontal axis to produce duplicates. Some images were discarded because they were nearly identical to the originals. In the end the experimental dataset consisted of 1677 images.

3.2 Model training

All YOLOv5 architectures in this experiment were trained using the Kaggle platform and the NVIDIA Tesla P100 GPU accelerator, which is freely available online via the Google Colab platform. This environment is particularly popular for data mining and computer vision research. To objectively compare the models, a Jupyter notebook was developed using the *ultralytics* library[1] to integrate with the YOLO model. The Jupyter notebook was reused for all 5 models included in the analysis and was therefore adapted to the Google Colab and Kaggle platforms. Also, the developed Jupyter notebook implements compression of outputs in ZIP format to expediate results download. The ability to resume data processing, load and store previously trained network parameters was implemented because of various usage restrictions of online platforms.

The developed Juypter notebook is freely available for non-commercial, educational and research purposes at: https://github.com/mhorvat/YOLOv5-models-comparison. We believe that the reported results and utilized real-world experimental settings are relevant to researchers who do not have access to powerful GPU workstations, especially junior early career researchers.

The experimental dataset was randomly divided in 3 subsets for training, evaluation and testing. This approach enables cross-validation and early stopping to prevent model overfitting. The relative sizes of the subsets were 85% (training), 10% (evaluation) and 5% (testing), respectively. The data was rotated during one training session using k-fold cross validation (k = 10). The cross-validation was stratified. However, the classes were imbalanced since 3.184 images were labelled "with mask", 716 as "without mask", and only 121 "mask worn incorrectly". In the worst case this could result in the neural network labelling all samples as "with mask" to minimize loss.

Table 2. Duration of training of YOLOv5 models in hours on the experimental Face Mask Detection dataset.

Model	Training duration [h]
YOLOv5n	1.17
YOLOv5s	0.83
YOLOv5m	1.83
YOLOv5l	2.83
YOLOv5x	8.67

Each model was trained for 300 epochs. The mini groups sizes were determined with *autobatch* function in *ultralytics* library. Patience property was set to 100 epochs to interrupt the training in the event of no progress. As could be seen in Table 2, training of the YOLOv5s model was the shortest. under an hour, while training of YOLOv5x was the longest and lasted almost nine hours. Generally speaking, the duration of training increases proportionally to the number of model parameters. However, due to the enabled early stopping feature, it is possible that training of more complex models will be shorter than for simpler models, as has happened with models *n* and *s*. The Google Colab has implemented a limit on free use (i.e. the stopping feature). In this regard, all Google Colab notebooks have an idle timeout of 90 minutes and an absolute timeout of 12 hours. If the user does not

1 https://github.com/ultralytics

interact with their Google Colab notebook for more than 90 minutes, the instance will automatically stop.

4 Results and discussion

The performance of the neural network is first determined by testing with a subset of training data that can indicate whether a particular model has sufficient learning ability to be used for this purpose, and with a subset of test data to determine the generalization ability of the network and how well it performs with the unknown input data. The classification experiment was performed independently on five YOLOv5 architectures (*n*, *s*, *m*, *l*, *x*) first on the training and then on the testing image subsets. An example of models' output is shown in Fig. 2.

Figure 2. Example of image classifications as outputs from the YOLOv5x model. Outputs from other YOLOv5 architectures in this study have the same format.

The results of models' performance assessment on training and testing subsets are given in Fig. 3. The most complex architecture YOLOv5x achieved the best results in the training subset, while models m and l performed slightly worse. This is quite expected, as is the fact that YOLOv5n performed the worst due to the correlation between the number of parameters and the ability to learn the network. Results on the training and testing subsets are presented to maintain the completeness of the report.

When testing with a set of unseen data (i.e. the testing subset), simpler neural networks can produce better results than more complex networks because of the ability to retrain the more complex networks. A smaller number of parameters alone can act as a form of regularization, giving a simpler network a greater ability to generalize.

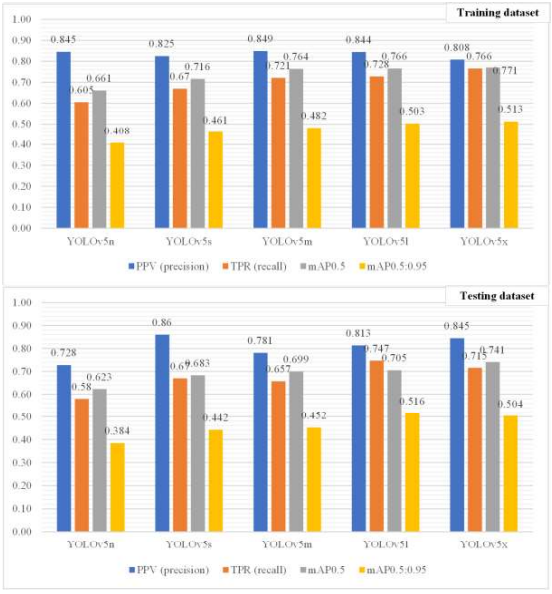

Figure 3. YOLOv5 (*n*, *s*, *m*, *l*, *x*) models' performance on the training subset (above) and testing data subset (below).

We see that more complex models are certainly more accurate, both in seen and unseen examples, but still, we cannot say that they are therefore better, especially if we take into account the everyday applications. In addition to accuracy itself, it is important, especially in the context of everyday applications, to consider the time it takes each model to process an unseen input example. The time it takes each model to process an image is given in Table 3.

A can be seen in Table 3, when choosing an optimal network architecture, it is necessary to make a trade-off between quality and speed. For more complex models to be competitive in terms of speed, they need stronger hardware, which may not always be available. For example, for a single-use device that automatically opens a door depending on whether a person is wearing a mask, it is not necessary to use a workstation when a device like the Raspberry Pi microcontroller with a camera could be used that can deliver nearly equal performance at much smaller cost.

Table 3. Duration of processing of a single random image from the experimental dataset. All values are expressed in milliseconds (ms). The shortest total processing time is in green, and the longest in red.

Model	Pre-processing	Processing	NMS	Total
YOLOv5n	0.2	5.1	2.5	7.8
YOLOv5s	0.5	5.0	2.0	7.5
YOLOv5m	0.3	11.2	1.8	13.3
YOLOv5l	0.6	17.8	2.0	20.4
YOLOv5x	0.8	32.7	1.8	35.3

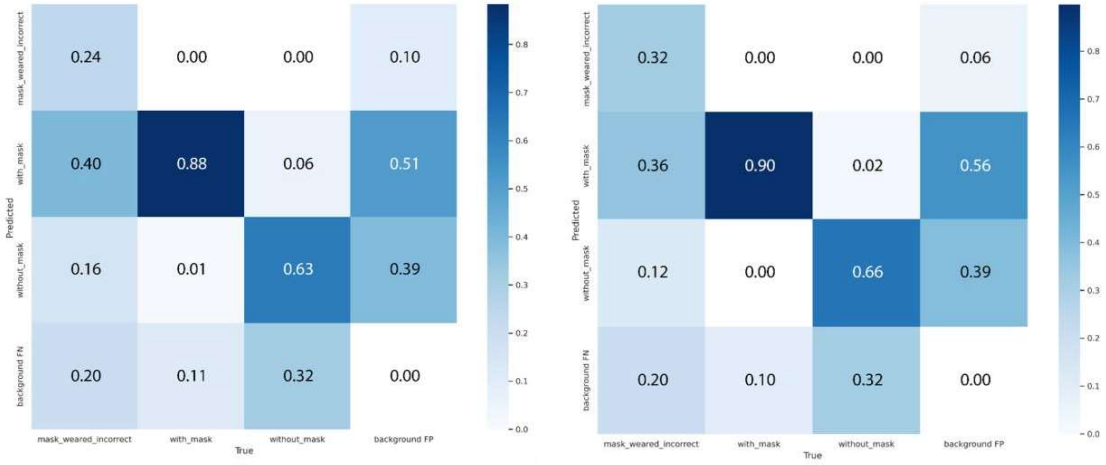

Figure 4. Confusion matrices for image classification with the most complex model YOLOv5x on the train (left) and test (right) subsets.

The YOLOv5x architecture achieved 88% TP (True Positive) for the class "with_mask" with the most examples in the training subset, and also 63% and 24% for the “without_mask” and "mask_weared_incorrect" classes with fewer data instances, respectively. In the testing, the YOLOv5x correctly classified 90% of “with_mask” instances, 66% “without_mask”, and 32% “mask_weared_incorrect” which is slightly better overall than in training. Nevertheless, such results on the testing subset may be explained by overfitting of the neural network during the training. Also, the results could be at least partially attributed to the previously mentioned problem of severely unbalanced class distribution. Class "mask_weared_incorrect" has the least images in the data set and examples belonging to this class in 52% of cases are falsely marked as "with_mask" which has the most instances in the dataset.

The results reported here and the YOLOv5 models implemented in a Jupyter notebook in the Google Colab development environment are freely available for educational and research purposes using the link provided above. Please cite this document when using the experimental results, software code, or conclusions in your work.

5 Conclusion

In the broader sense, the main advantage of YOLOv5 is twofold. First, it is the integration of advances from other research areas in computer vision theory that improve YOLOv5 object recognition performance. Second, they are made available to ML and data scientists as a collection in the PyTorch framework, which can be easily integrated and used in modern cloud-based application development environments and other software packages such as NumPy.

The latest YOLO versions v6 and v7 show even better performance than v5, especially in detecting small objects in densely packed environments. However, these versions have only recently been released and should be tested more thoroughly in various settings and with different datasets.

The YOLOv5 algorithm has significant improvements over its predecessor, YOLOv3, making it easier to train with a range of written scripts and simplifying the later application of a trained image processing network. In addition, it offers a full range of architectures of varying complexity enabling a range of new everyday applications and indicating that accuracy is not the only item to pay attention to when choosing a network. The YOLOv5 improves and facilitates the application of trained neural networks in real-time video signal processing applications such as crowd people counters (Doljanin et al., 2021; Sukkar, Kumar & Sindha, 2021), or as a part of an intelligent decision support system for monitoring student behavior that can assist teachers in optimizing personalized education during online courses (Horvat & Jagušt, 2020), or for example, in an automated system that selectively opens doors to public facilities only to persons properly wearing a medical mask.

In a nutshell, the versatile YOLOv5 algorithm can train a neural network for various practical applications, one of which is demonstrated in this paper. Moreover, unlike its predecessors, YOLOv5 improves the application of trained neural networks in video processing applications. It allows such applications to run on weaker devices, which has been challenging to do so far.

The YOLO architecture is ready to use out of the box, which is especially useful to junior engineers and researchers without much experience in machine learning and computer vision. But to improve object recognition performance without changing the algorithm, it would be advisable first to improve the

understanding of the scene. For example, classification performance could be improved by disregarding unnecessary parts of an image and selecting only genuine regions of interest before importing the image into the YOLO pipeline. In this regard, it is highly advisable to use ontologies, formal knowledge representation methods, and automated reasoning services to achieve a semantically rich description of the scene and an understanding of the concepts in the image and their mutual functional relationships (Horvat, 2013; Horvat, Grbin & Gledec, 2013a; Horvat, Grbin & Gledec, 2013b).

Finally, another important general point in practical applications of AI and DL algorithms such as the YOLO must be taken into consideration, and that is reliability or trust. Existing AI methods, if adequately trained, are very effective. Still, their output is very often hard to interpret and understand by the end-user, depending on the AI system's decisions. Therefore, in an Explainable AI (XAI) approach, an algorithm needs to provide the user with an explainable decision (Nott, 2017). Explainable AI output aims to provide the user with additional information that supports the decision made by an AI system. This is usually a set of key features that lead to the system's decision. Providing XAI explanations to the end user increases confidence and understanding of the system. To take full advantage of the YOLO family of algorithms, including future updated versions, it is vital to use them to provide a comprehensive explanation of why a particular classification or decision was made.

References

Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

Chiang, D. (2020). AIZOOTech. *FaceMaskDetection*. Retrieved from https://github.com/AIZOOTech/FaceMaskDetection

Doljanin, D., Pranjić, L., Jelećević, L., & Horvat, M. (2021). Adaptive intelligent agent for e-learning: First report on enabling technology solutions. In *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)* (pp. 1690-1694). IEEE.

Du, Y., Pan, N., Xu, Z., Deng, F., Shen, Y., & Kang, H. (2021). Pavement distress detection and classification based on YOLO network. *International Journal of Pavement Engineering*, *22*(13), 1659-1672.

Fang, W., Wang, L., & Ren, P. (2019). Tinier-YOLO: A real-time object detection method for constrained environments. *IEEE Access*, *8*, 1935-1944.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, *37*(9), 1904-1916.

Horvat, M. (2013). *Generation of multimedia stimuli based on ontological affective and semantic annotation* (Doctoral dissertation, Zagreb: University of Zagreb Faculty of Electrical Engineering and Computing).

Horvat, M., Grbin, A., & Gledec, G. (2013). *WNtags: A web-based tool for image labeling and retrieval with lexical ontologies*. Frontiers in artificial intelligence and applications, 243, 585-594.

Horvat, M., Grbin, A., & Gledec, G. (2013). Labeling and retrieval of emotionally-annotated images using WordNet. *International Journal of Knowledge-based and Intelligent Engineering Systems*, *17*(2), 157-166.

Horvat, M., & Jagušt, T. (2020). Emerging Opportunities for Education in the Time of COVID-19-Adaptive e-Learning Intelligent Agent Based on Assessment of Emotion and Attention. In *Central European Conference on Information and Intelligent Systems* (pp. 203-210). Faculty of Organization and Informatics Varazdin.

Ieamsaard, J., Charoensook, S. N., & Yammen, S. (2021, March). Deep learning-based face mask detection using yolov5. In *2021 9th International Electrical Engineering Congress (iEECON)* (pp. 428-431). IEEE.

Jiang, X., Gao, T., Zhu, Z., & Zhao, Y. (2021). Real-time face mask detection method based on YOLOv3. *Electronics*, *10*(7), 837.

Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. (2022). A Review of Yolo algorithm developments. *Procedia Computer Science*, *199*, 1066-1073.

Jocher, G. (2020). YOLOv5. Retrieved from https://github.com/ultralytics/yolov5

Jocher, G. (2020). YOLOv5: Train Custom Data. Retrieved from https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data/

Kim, D., Park, S., Kang, D., & Paik, J. (2019, September). Improved center and scale prediction-based pedestrian detection using convolutional block. In *2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)* (pp. 418-419). IEEE.

Krišto, M., Ivasic-Kos, M., & Pobar, M. (2020). Thermal object detection in difficult weather conditions using YOLO. *IEEE access*, *8*, 125459-125476.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*. .

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham.

Maranhão, A. (2020). Face Mask Detection. Retrieved from https://www.kaggle.com/andrewmvd/face-mask-detection

YOLOv6: a single-stage object detection framework dedicated to industrial applications. (2022). Retrieved from https://github.com/meituan/YOLOv6

Nelson, J., & Solawetz, J. (2020). Responding to the Controversy about YOLOv5. *Roboflow Blog*. Retrieved from https://blog.roboflow.com/yolov4-versus-yolov5/

Nott, G. (2017). Explainable artificial intelligence: Cracking open the black box of AI. *Computer world*, *4*.

Padilla, R., Netto, S. L., & Da Silva, E. A. (2020, July). A survey on performance metrics for object-detection algorithms. In *2020 international conference on systems, signals and image processing (IWSSIP)* (pp. 237-242). IEEE.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7263-7271).

Redmon, J., & Farhadi, A. (2018). Yolo: Real-time object detection. *Pjreddie. com*. Retrieved from https://github.com/pjreddie/darknet/wiki/YOLO:-Real-Time-Object-Detection

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Sharma, V. (2020). *Face mask detection using yolov5 for COVID-19* (Doctoral dissertation, California State University San Marcos).

Sukkar, M., Kumar, D., & Sindha, J. (2021, July). Real-Time Pedestrians Detection by YOLOv5. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (pp. 01-06). IEEE.

Solawetz, J. (2020). Yolov5 new version-improvements and evaluation. *Roboflow. Seach date*. Retrieved from https://blog.roboflow.com/yolov5-improvements-and-evaluation/

Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*.

Wang, W., Xie, E., Song, X., Zang, Y., Wang, W., Lu, T., ... & Shen, C. (2019). Efficient and accurate arbitrary-shaped text detection with pixel aggregation network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 8440-8449).

Yang, G., Feng, W., Jin, J., Lei, Q., Li, X., Gui, G., & Wang, W. (2020, December). Face mask recognition system with YOLOV5 based on image recognition. In *2020 IEEE 6th International Conference on Computer and Communications (ICCC)* (pp. 1398-1404). IEEE.