Performance Analysis of YOLO-NAS SOTA Models on CAL Tool Detection

Muhammad Adil Raja, Róisín Loughran, Fergal McCaffery
Regulated Software Research Center (RSRC)
Dundalk Institute of Technology (DkIT)
Dundalk, Ireland
adil.raja, roisin.loughran, fergal.mccaffery@dkit.ie

***Abstract*—Every now and then, we witness significant improvements in the performance of Deep Learning models. A typical cycle of improvement involves enhanced accuracy followed by reduced computing time. As algorithms get better at their job, it is worthwhile to try to evaluate their performance on problems that are affected by them. Computationally intense problems, such as object detection for Computer Aided Laparoscopy (CAL), can benefit from such improvements in such technologies. Recently a new set of variants of You Look Only Once (YOLO) models based on Neural Architecture Search (NAS) technique have been released. Deci, the enterprise behind this new development, touts a much better performance both in terms of accuracy as well as computational efficiency.**

In this paper, we have analyzed the performance YOLO-NAS on a well-known benchmark dataset related to CAL. We found that the performance of all the NAS-based YOLO was inferior as compared to other State-of-the-Art (SoTA) YOLO models. We compare our results against the YOLOv7 model too.

***Keywords*—Computer-aided laparoscopy, cholecystectomy, object detection, deep learning, convolutional neural networks.**

I. Introduction

Computer Vision (CV) algorithms for object detection have advanced significantly since the initial implementation of Convolutional Neural Networks (CNNs). Inception of the You Look Only Once (YOLO) algorithm was a significant stride at object detection because of the way the algorithm works. It substantially reduced the processing time for computing the bounding boxes as well as the labels for objects present in an image. Subsequent versions of the algorithm were aimed at further enhancing the accuracy as well as computational efficiency of the algorithm. Thus, amendments were made to the structure of the model as well as to the central algorithm over the years to address these issues. In 2022 and 2023 respectively, both version seven and version eight of the algorithm were released. Most recently, Deci released new models for the algorithm based on Neural Architecture Search (NAS). They have released a small, a medium-sized, and a large model that they developed using NAS. On their website, as well as on other online outlets, they have reported a much superior performance both in terms of accuracy as well as computational efficiency of the NAS-based models. Fig. 1 shows a snapshot of the performance of the model. This picture was taken from the GitHub webpage of Deci-AI that they have dedicated for the YOLO-NAS's software repository [1]. It can be seen that all variants of YOLO-NAS outperform all other State-of-the-Art (SoTA) models both on accuracy as well as latency.

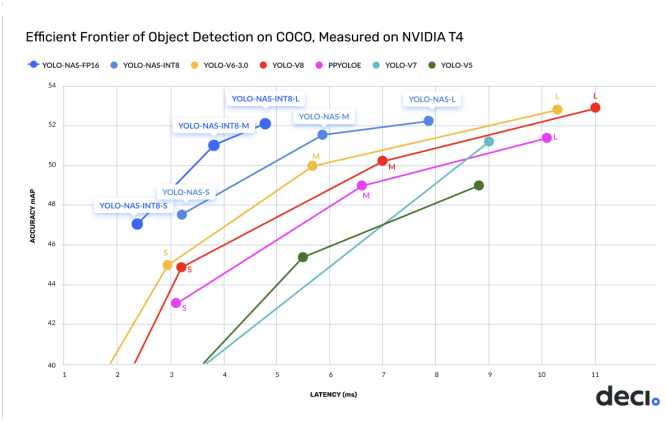


Fig. 1: Efficient frontier detection on COCO.

However, when we compared the performance of all the variants of YOLO-NAS (in this paper) on a custom dataset related to Computer Aided Laparoscopy (CAL) of cholecystectomy, we found that the performance of the models was inferior as compared to other SoTA models.

This paper is part of a series of articles that explore the efficiency of various SoTA CV algorithms and models at predicting CAL instruments used in robotic surgical practices. CAL can have many benefits for the future of surgery. And CV algorithms have a central role to play in this as they can allow for real-time and accurate tagging of surgical tools as well as human anatomical structures in the operation theatre.

The rest of this paper is organized as follows: In section II, we provide a brief background of object detection as well as the YOLO algorithm. Section IV introduces CAL and the role of CV in it. Section V presents our experimental details. In section VI, we have reported the results of our study. Finally, section VII concludes this paper.

This research was funded by the Technological University Transfer Fund (TUTF) of the Higher Education Authority (HEA) and DkIT.

II. Object Detection and the YOLO Algorithm

Object detection is the craft of tagging objects in an image and assigning labels to them depending on their class. The difference between detection and classification is quite subtle but distinct. In classification, the algorithm is only capable of giving a prediction about the class or category of the objects present in the image. The detection algorithm goes a step further by providing the precise location of the object(s) in the image.

Object detection was advanced significantly with the advent of CNNs along with advancements in Graphical Processing Unit (GPU) hardware. At the heart of these novel Artificial Neural Networks (ANNs) lies the idea of convolution. Convolution by itself is not a new idea, it is well-established in the field of signal processing. Convolution involves the computation of a cross-correlation between two signals. One among these is normally a signal under test. The other one can be thought of as a template or a target signal that we seek to find in the signal under test. When this cross-correlation is performed over the entire gamut of the signal under test, using the target signal, the operator is termed convolution. Although convolution has been well-known to the signal processing community for a long time, its adoption in CV was made possible only due to the arrival of GPU hardware.

Initial detection algorithms, although inspiring, were quite slow when compared with the real-time computation demands of real-world CV applications. In recent years, a newer type of object detection, YOLO, has appeared as an open-source software [2]. This algorithm solves the problem of object detection by treating it as a regression problem. It must be remembered that the detection problem was mostly a classification problem in a classical sense. The reason, as suggested earlier, is due to the requirement on the algorithm to predict the class of the objects present in the image.

In treating the detection problem as a regression problem, the YOLO algorithm divides the image into a grid of cells. It initially generates a certain number of bounding boxes in the hope of finding something befitting for the bounding box of the target image. From among these boxes, it chooses those that closely overlap the boxes of the objects in the image. It then attempts to figure out if an object is present in a given grid cell. Once it has done that, it predicts the classes of the objects in the bounding boxes. If this is the case, the generated bounding boxes of this cell are matched to the target bounding box of the image and a prediction about the class of object present in the cell is made. Two things should be noted. One is that it computes the discrepancy between the prediction of the bounding box as well as the classes with the corresponding value of the target in the sense of a regression problem. The loss function is also defined in these terms. In simple terms, the loss function measures the difference between the prediction of the algorithm and the actual target. The second is that it does both the prediction of the box as well as the class(es) in one pass, and hence its name. In completing its whole job in one pass, the algorithm saves a substantial amount of time

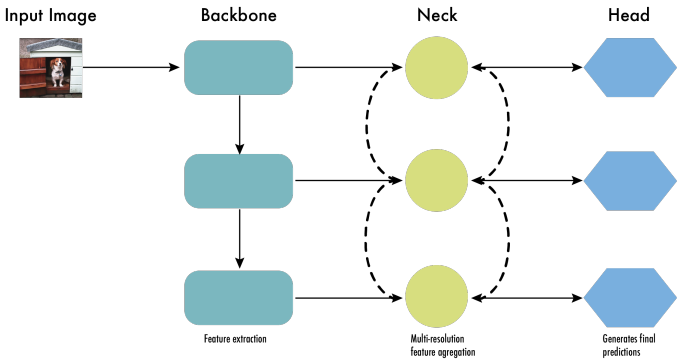


Fig. 2: High level architecture of the YOLO model.

as compared to its predecessors. Recent developments of the algorithm have shown impressive computational efficiency [3].

YOLO has come a long way since its inception. Currently, the algorithm has eight versions that are hosted at different websites. The current architecture of the models is given in Fig. 2 showing a high-level view of the architecture of a typical YOLO model [4]. The whole architecture can be divided into three different blocks. These are, namely, a backbone, a neck, and a head. The backbone acts as a feature extractor and computes feature maps to be processed by subsequent layers. The neck connects the backbone to the head. It concatenates feature maps from the various layers of the backbone and sends them as inputs to the head. The head is responsible for processing the input from the neck and to compute objectness scores, bounding boxes, and class probabilities.

III. Neural Architecture Search – The Need and Benefits

ANNs were invented to solve formidable real-world problems, that do not lend solutions using simplistic techniques. There are numerous practical problems of such a kind that their solutions or mathematical formulations do not lend themselves easily to human imagination. Hence, when it is impossible to conceptualize a problem visually and it cannot be formalized mathematically, ANNs lend themselves as handy tools to solve them. The way ANNs work is that they dedicate ample resources in terms of a large number of small non-linear functions. By doing this, they try to capture the non-linear trends that underlie the problem at hand.

In traditional ANN, structure of the network is generally hand-picked. Normally it is a hit-and-trial pursuit in which the practitioner experiments with a few randomly chosen topologies until the results get better. Thus, there is limited intuition or mathematical insight in designing the model. The practitioner settles for what works best in terms of the structure. The downside of this approach is that a possibly wider search space involving neural structures is not explored appropriately. The practitioners have to work with whatever they find better in a limited period. Due to this, they can finally end up with a model that produces sub-optimal results.

The quest for automating the process of finding an adequate structure for an ANN, given a computational problem, is at

least three decades old. The earliest academic literature that can be found about the enterprise of NAS is cited in [5], [6]. The central idea is to employ a meta-heuristic algorithm to search for a structure that will lead to near-optimal results. Commonly used meta-heuristics involve Evolutionary Algorithms (EAs) and other nature-inspired schemes for developing computational intelligence.

NAS was initially only extensively applied for the case of conventional back-propagation shallow ANNs. In recent years, owing to the rapid advancements in the GPU hardware, NAS schemes have also been developed for deep learning ANNs, including CNNs.

The YOLO algorithm, as well as any resulting models, belong to the family of CNNs that treat object detection as a regression problem [2].

In doing this, the algorithm assumes the availability of ample computing resources at the practitioner's end. Now that the algorithm and the models have achieved considerable maturity in object detection, the hope is to seek further improvement through NAS.

Deci's versions of YOLO-NAS models are based on their proprietary Automated Neural Architecture Construction (autoNaC) technology. However, according to Deci's own description of it, autoNaC is a very miniature version of the overall NAS. In essence, autoNaC reduces an already trained model to one of a certain discrete number of designs.

IV. CAL AND CV

Laparoscopy allows a surgeon to have access to operate on a human body without making large incisions. For many surgical purposes, laparoscopy can be preferred over open surgery. It imparts a much smaller incision, decreases the recovery time, reduces blood loss, and affects the healing nicely [7]. The traumatic effects of traditional surgery are diminished due to it. The chances of catching infections are also reduced. However, with its numerous advantages certain challenges are also associated. Poor hand-eye coordination and a narrow field of view are major among these. Instrument detection, real-time body part information, surgical phase revelation, and 3D pose estimation can aid a surgeon tremendously while performing CAL. These tasks can become extremely difficult because of the presence of smoke, blockages, blood, shadows, reflections, blurring of movement, cleaning gauze, and complicated background surfaces [8]. All of these challenges require the development of more sophisticated CV systems for tracking tools for CAL.

Deep learning and CNNs brought a revolution in CV during the past decade [9]. The cheap availability of High Performance Computing (HPC) devices and the widely available GPU power have paved the way for agile development of CV applications. Medical imaging and CAL have also benefited from this. Once capable of solving classification problems only, CNNs were later modified and enhanced to achieve the capacity for object detection [10]. Due to the ability of object detection CAL has come a long way. As algorithms get better, in addition to detecting objects with increasing accuracy, their computational efficiency also improves.

TABLE I: The Number of Instances of Various Tools in the Dataset

Instrument	Training	Validation	Test
Grasper	707	420	293
Bipolar	215	120	95
Hook	165	79	64
Scissors	197	107	84
Clipper	220	116	64
Irrigator	228	173	84
Specimen Bag	241	138	96

V. EXPERIMENTAL DETAILS

The well-known m2cai16-tool-locations dataset [11] was employed in this research. The dataset contains 2,811 labeled images of surgical tools. The names of these tools are grasper, bipolar, hook, scissors, clipper, irrigator, and specimen bag. Fig. **??** shows pictures of these tools including the bounding boxes [1].

An average of 1.2 labels per frame are present in the dataset. The dataset is split into 50%, 30%, and 20% for training, validation and testing respectively. The overall distribution of the data is shown in Table I.

All three variants of the YOLO-NAS algorithm were trained using this data.

The training was conducted using the Kay supercomputer provided by the Irish Center for High end Computing (ICHEC). The computer on which training was performed has two NVIDIA Tesla V100 16GB PCIe (Volta architecture) GPUs. There are 5,120 CUDA cores and 640 Tensor Cores on each of the GPUs. Training of the smallest model (i.e. YOLOv8n) was conducted using an NVIDIA GeForce RTX 2060 with Max-Q Design, having 5927MiB of on-chip memory. Inference results for all the models were carried out using this later GPU.

Default values for the hyperparameters for training were used. The number of training epochs was set to 200. Moreover, since the image sizes in the dataset are 596×334 pixels, the image size for training, validation, and inference was set to 596 pixels, as opposed to the default of 640.

VI. RESULTS

In this section, we report the results of our training of all three variants of YOLO-NAS models, namely, small, medium, and large. The results are related to object detection of surgical tools used in CAL of cholecystectomy. Fig. 4 shows the confusion matrix related to the trained model evaluated on the unseen test data. The results are reported in terms of percentages rounded to the nearest integer values.

Fig. 5 shows Precision (P)-Recall (R) curves. These curves show the tradeoff between precision and recall for different values of threshold. A high precision is related to a low false

[1]This figure was taken from the website of the original data repository: https://ai.stanford.edu/ syyeung/tooldetection.html

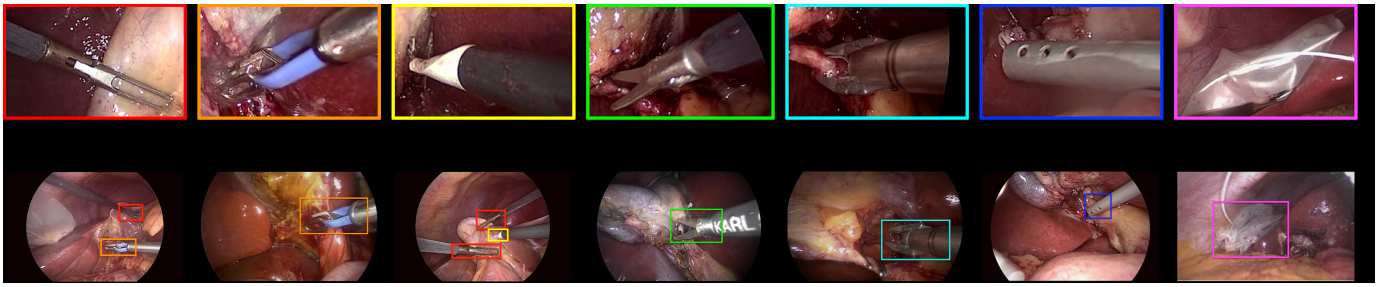


Fig. 3: Different tools present in the dataset. Their bounding boxes are shown in the lower row.

positive rate. Whereas a high recall is related to a low false negative rate. The area under the curve tells us about the degree of precision and recall. A large area is reminiscent of high precision and high recall.

Fig. 6 shows pictorial results for the curious reader. The pictures in the left column show the ground truth labels of different batches of data as well as the target bounding boxes. Whereas the pictures in the right column show the labels as well as the bounding boxes predicted by the model. These figures are related to the unseen test data.

Table II shows the performance of various YOLO-NAS models in terms of different metrics. The chosen metrics are P, R, mean Average Precision (mAP), and F1 at a fifty percent confidence level.

TABLE II: Performance of YOLOv8x on Unseen Test Data

Class	P@0.5	R@0.5	mAP@0.50	F1@0.50
YOLO-NAS-s	0.107	0.973	0.915	0.192
YOLO-NAS-m	0.105	0.978	0.917	0.187
YOLO-NAS-l	0.089	0.972	0.915	0.163

Table III shows a comparison of different models in terms of mAP@50 metric. Again, the results are related to the unseen test data. This table shows results viz a viz each tool separately. The table also shows the performance of the YOLOv7 and YOLOv8n, which is the smallest variant of YOLOv8. It is worth mentioning that all YOLO-NAS models performed dismally in detecting the bipolar.

TABLE III: Comparison of Different Models in Terms of mAP@50

Class	NAS-s	NAS-m	NAS-l	v7	v8n
All tools	0.915	0.917	0.915	0.957	0.959
Grasper	0.712	0.755	0.819	0.917	0.915
Bipolar	0.131	0.186	0.024	0.956	0.970
Hook	0.930	0.936	0.966	0.987	0.995
Scissors	0.913	0.906	0.921	0.936	0.972
Clipper	0.964	0.920	0.915	0.988	0.961
Irrigator	0.888	0.936	0.817	0.944	0.942
SpecimenBag	0.912	0.819	0.940	0.969	0.956

Finally, Figs. 7 and 8 show the confusion matrices for YOLOv7 and YOLOv8n also, respectively.

VII. Conclusions

In this paper, we have presented results related to tool detection of CAL instruments. Particularly, data related to cholecystectomy was used to train all three variants of YOLONAS. Seven different types of instruments were present in the data. Results of the trained object detection model have been reported in this paper, which are dismal as compared with other SoTA models. In particular, the models were compared with YOLOv7 and YOLOv8n and it was found that these later models had much better performance on the unseen test data.

References

[1] S. Aharon, Louis-Dupont, Ofri Masad, K. Yurkova, Lotem Fridman, Lkdci, E. Khvedchenya, R. Rubin, N. Bagrov, B. Tymchenko, T. Keren, A. Zhilko, and Eran-Deci, "Super-gradients," 2021. [Online]. Available: https://zenodo.org/record/7789328

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[3] J. Terven and D. Cordova-Esparza, "A comprehensive review of yolo: From yolov1 and beyond. arxiv 2023," *arXiv preprint arXiv:2304.00501*.

[4] H.-B. Le, T. D. Kim, M.-H. Ha, A. L. Q. Tran, D.-T. Nguyen, and X.-M. Dinh, "Robust surgical tool detection in laparoscopic surgery using yolov8 model," in *2023 International Conference on System Science and Engineering (ICSSE)*. IEEE, 2023, pp. 537–542.

[5] F. Gruau, "Automatic definition of modular neural networks," *Adaptive behavior*, vol. 3, no. 2, pp. 151–183, 1994.

[6] X. Yao, "A review of evolutionary artificial neural networks," *International journal of intelligent systems*, vol. 8, no. 4, pp. 539–567, 1993.

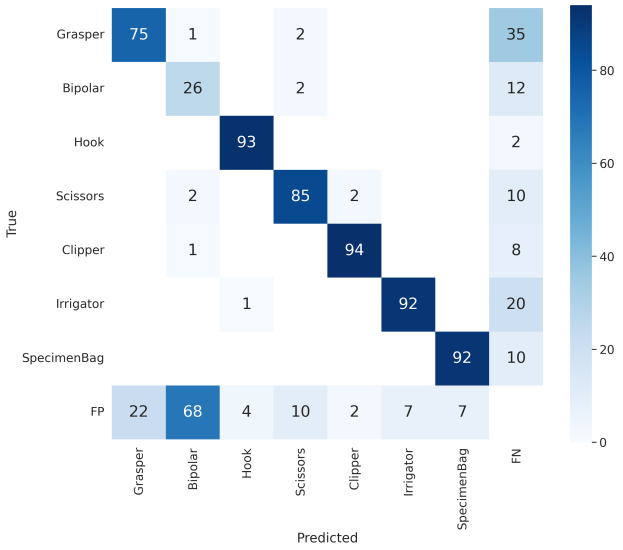
[7] G. Ietto, F. Amico, G. Pettinato, V. Iori, and G. Carcano, "Laparoscopy in emergency: why not? advantages of laparoscopy in major emergency: a review," *Life*, vol. 11, no. 9, p. 917, 2021.

[8] M. K. Hasan, L. Calvet, N. Rabbani, and A. Bartoli, "Detection, segmentation, and 3d pose estimation of surgical tools using convolutional neural networks and algebraic geometry," *Medical Image Analysis*, vol. 70, p. 101994, 2021.

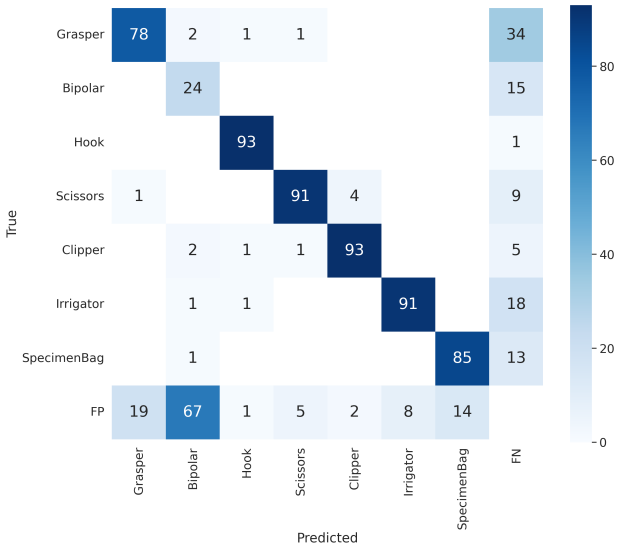
[9] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks," *Pattern recognition*, vol. 77, pp. 354–377, 2018.

[10] K. Ragland and P. Tharcis, "A survey on object detection, classification and tracking methods," *Int. J. Eng. Res. Technol*, vol. 3, no. 11, pp. 622–628, 2014.

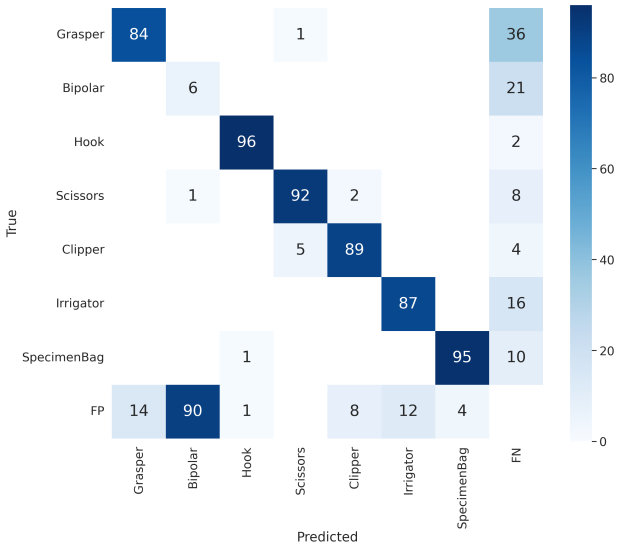
[11] A. Jin, S. Yeung, J. Jopling, J. Krause, D. Azagury, A. Milstein, and L. Fei-Fei, "Tool detection and operative skill assessment in surgical videos using region-based convolutional neural networks," *IEEE Winter Conference on Applications of Computer Vision*, 2018.



(a) Small

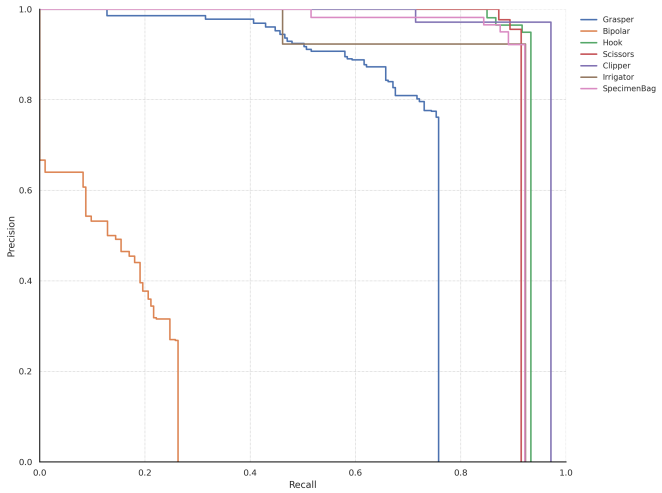


(b) Medium

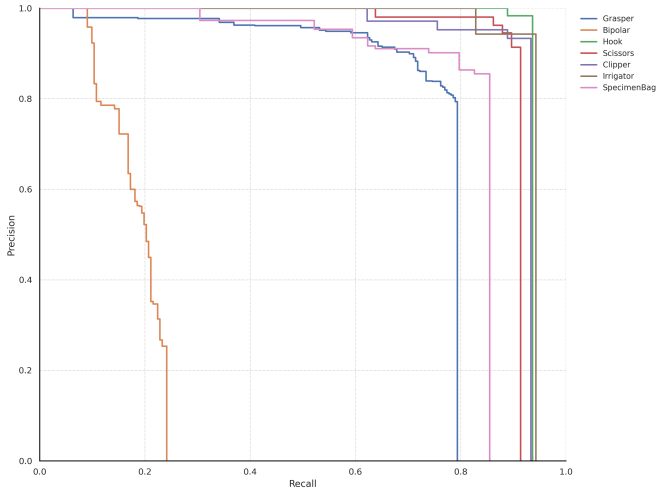


(c) Large

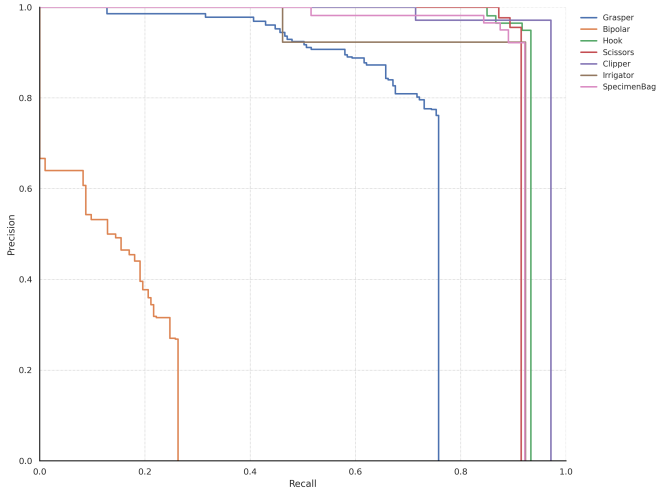
Fig. 4: Confusion matrices related to the unseen test data for small, medium, and large YOLO-NAS models.



(a) Small



(b) Medium



(c) Large

Fig. 5: Plots of P versus R on unseen test data for small, medium and large YOLO-NAS models.

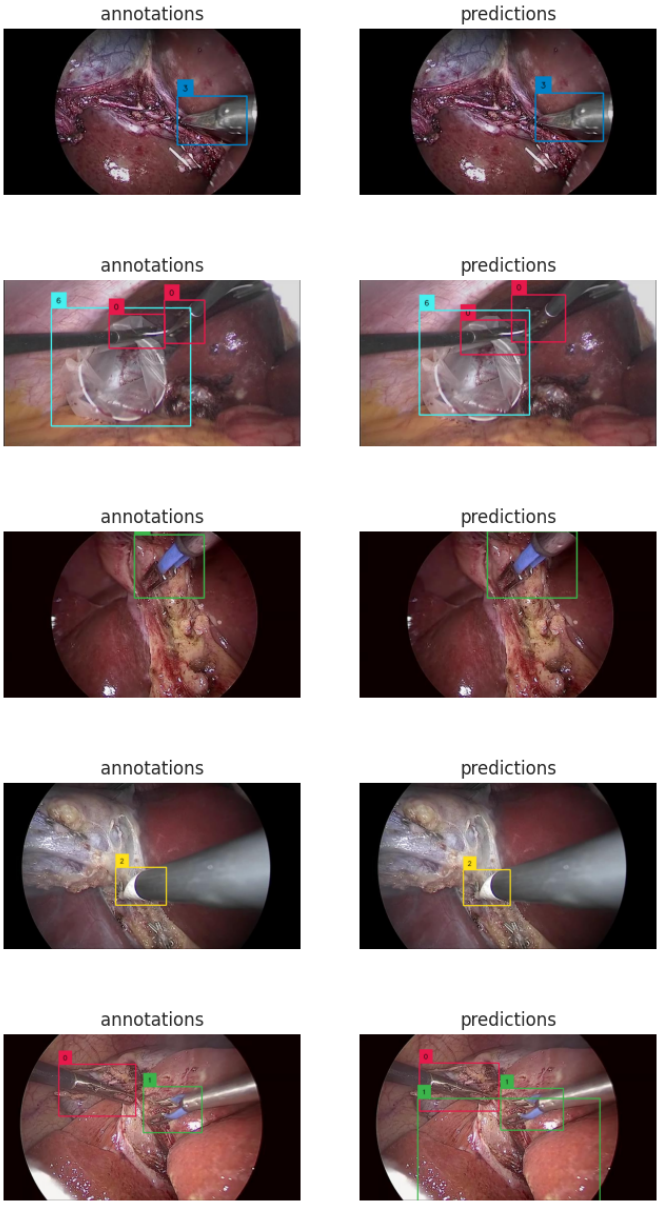


Fig. 6: Plots of annotations versus predictions on unseen test data for YOLO-NAS-L model.

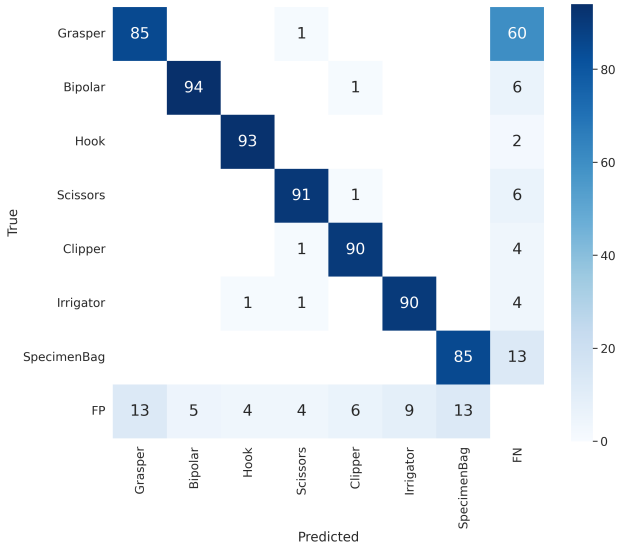


Fig. 7: Confusion matrix for YOLOv7 on unseen test data.

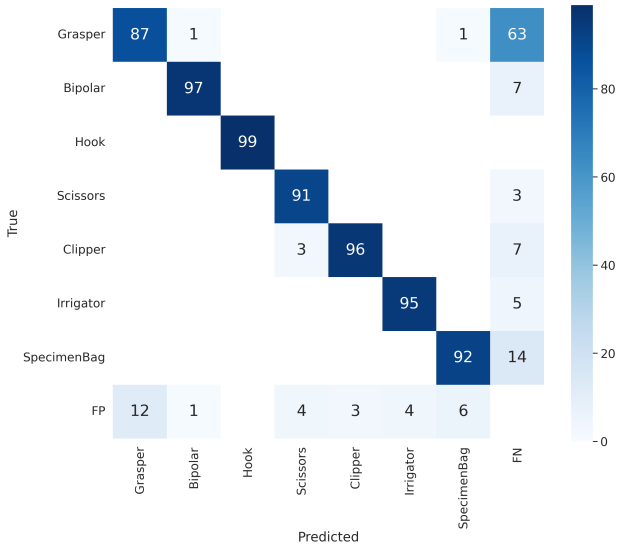


Fig. 8: Confusion matrix for YOLOv8n on unseen test data.