Surgical Tool Detection

N V Sai Likhith
I.MTech CSE
Internation Institute of Information Technology
Bangalore, India
VenkataSai.Likhith@iiitb.ac.in

***Abstract*—This document is a report of the Project of Surgical Tool Detection done under *Prof. Neelam Sinha, IIITB* and *Sai Pradeep, PhD, IIITB* in the Spring 2023.**

I. Aim

The objectives of the project can be summarized as follows:

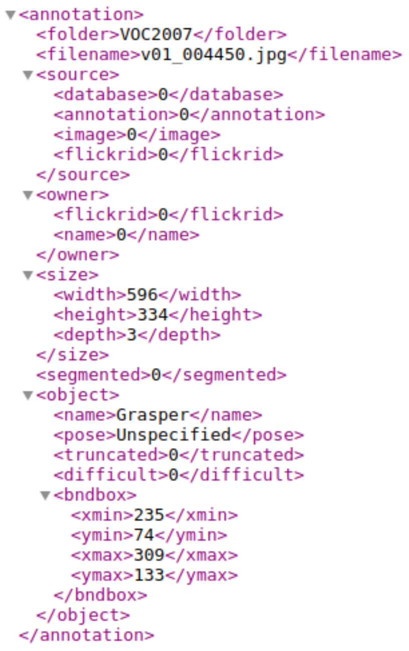
- Understanding the m2cai-16 tool locations dataset
- Understanding the evolution of YOLO Architectures from v1 to v7
- Training the YOLOv7 model on the given dataset and getting mAP results on the same

II. Dataset Details

The dataset *m2cai16 tool locations* [1] is an extension of *m2cai16 tools* dataset with spatial locations. *m2cai16 tool* dataset contains surgical frames along with binary annotations.

The dataset developers annotate all frames of *m2cai16 tool* datset containing just one tool, and increase the number of annotated instances per tool class by additionally labeling frames with two and three tools

The annotations of the dataset will be provided in an XML format as shown in Figure 1.



```
▼<annotation>
   <folder>VOC2007</folder>
   <filename>v01_004450.jpg</filename>
  ▼<source>
     <database>0</database>
     <annotation>0</annotation>
     <image>0</image>
     <flickrid>0</flickrid>
   </source>
  ▼<owner>
     <flickrid>0</flickrid>
     <name>0</name>
   </owner>
  ▼<size>
     <width>596</width>
     <height>334</height>
     <depth>3</depth>
   </size>
   <segmented>0</segmented>
  ▼<object>
     <name>Grasper</name>
     <pose>Unspecified</pose>
     <truncated>0</truncated>
     <difficult>0</difficult>
    ▼<bndbox>
       <xmin>235</xmin>
       <ymin>74</ymin>
       <xmax>309</xmax>
       <ymax>133</ymax>
     </bndbox>
   </object>
 </annotation>
```

Fig. 1. XML annotation for an image

III. Literature Survey

In this project, I thoroughly reviewed the official papers of YOLOv1 [2], YOLOv2 [3], YOLOv3 [4], and YOLOv7 [5] to gain a comprehensive understanding of their advancements. To implement YOLOv7 on the dataset, I relied on the code shared by WangYu on his GitHub repository. In the following discussion, I will brief the working of YOLOv7.

A. YOLOv1

This is the very first version of the YOLO models. It is based on the architecture of GoogLeNet. It is a cascade of convolutional layers interleaved with MaxPool. The cascade ended with two fully connected layers.

Within this architecture, the original image undergoes a transformation into a grid with dimensions SxS. Within each cell of the grid, the network predicts B bounding boxes, accompanied by a confidence score indicating the likelihood of an (any) object's presence within each box. Additionally, each cell also provides predictions for the class probabilities of C classes. So each cell gives a vector of size 5B + C as the output.

To obtain the most accurate bounding box, it is crucial to address the issue of multiple bounding boxes being generated by adjacent cells, leading to redundancy. To tackle this, a technique called non-maximal suppression is employed. Here's how it works: First, all the bounding boxes associated with a particular class are gathered from the image. Bboxes with confidence scores below a specified threshold are eliminated. Then, a pairwise comparison using Intersection over Union (IoU) is performed. If the IoU between two compared bounding boxes exceeds 0.5, the bbox with the lower confidence score is discarded. Otherwise, both bboxes are retained. Through this process, similar and redundant bounding boxes are gradually reduced, ensuring optimal results.

The loss function used can be found at Figure 2. The first term accounts for the error in predicting the coordinates of the object's center. The second term quantifies the discrepancy in estimating the dimensions of the bounding box. The third term relates to the classification of the object's class. The fourth term captures the classification loss when the object is absent. Lastly, the fifth term represents the loss associated with the probabilities of detecting any object within the bounding box.

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

Fig. 2. Loss Function of YOLOv1

Despite the advantage in the speed of this version, it has its own limitations:

1) Restrictions in the form of 2 bboxes and one class of objects per cell. This means that a bunch of small objects are less recognized.
2) Several successive downsamples of the original image lead to not so high accuracy.
3) Loss is designed in such a way that it equally penalizes errors on large bboxes and small ones. The authors made an attempt to compensate for the effect by taking the root of the size, but this did not eliminate the effect completely.

B. *YOLOv7*

After several updates to the architecture, state-of-the-art (SOTA) results were achieved with YOLOv7. Notably, they incorporated trainable bag-of-freebies techniques such as model re-parameterization and model scaling. These enhancements aimed to improve accuracy while minimizing the impact on training cost. Additionally, a novel label assignment method was introduced, which involved the introduction of auxiliary heads. The outputs of these auxiliary heads were combined with the main head to obtain the labels for object detection, thereby contributing to the overall loss calculation. This approach helped further optimize the performance of YOLOv7.

IV. Methodology

I utilized the methodology of YOLOv7 and employed the code available on WongKinYu's GitHub repository [6] for my implementation. WongKinYu had originally trained the model on the MS COCO dataset. To adapt my own dataset, which contained annotations in XML format from the m2cai16 tool locations dataset, I converted them to the TXT format used in the MS COCO dataset. Subsequently, I formatted my dataset to align with WongKinYu's implementation and proceeded to train the model using the prepared training set.

The conversion from XML format (Figure 1) to TXT format `<class> <x1> <y1> <x2> <y2>` is done using a converter function that extracts the values from `object` column of XML annotation into the above mentioned TXT format.

Once the annotations are ready as required by WongKinYu's implementation, then the model can be trained on our dataset in CUDA environment. I had run it for 300 iterations and acheived appreciable results. Detailed Descussion will be done in the next Section.

V. Results

A. *Confusion Matrix*

The confusion matrix is a metric that offers an overall measure of accuracy for the model. It enables us to determine four essential evaluation metrics for each class: True Positive, True Negative, False Positive, and False Negative. These metrics play a crucial role in calculating various performance measures such as F1-score, mAP (mean Average Precision), and others. The rows denote the predicted class. and columns denote the original class. Look at Figure 3.

1) True Positive: Out of all the **positively** predicted objects, how many are **truly** positive.

2) True Negative: Out of all the **negatively** predicted objects, how many are **truly** positive.

3) False Positive: Out of all the **positively** predicted objects, how many are **not** positive.

4) True Negative: Out of all the **negatively** predicted objects, how many are **truly** negative.

Note that, by positive I mean the class of consideration and negative means set of all the other classes.

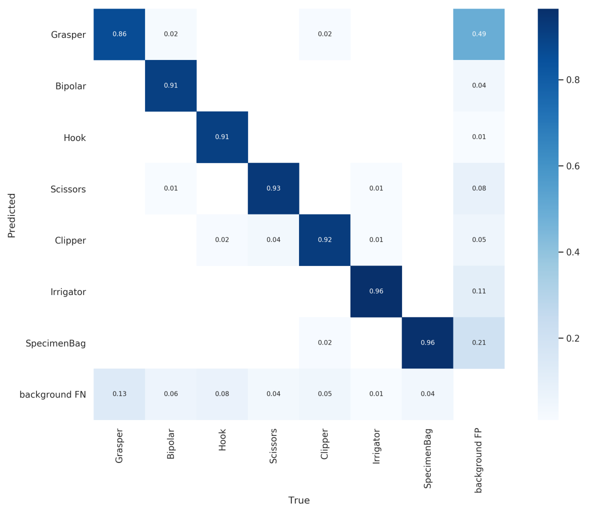


Fig. 3. Confusion Matrix

B. *F1-Score*

We need to define Precision and Recall as follows:
Precison: Within everything that has been predicted as a positive, precision counts the percentage that is correct.
Recall: Within everything that actually is positive, how many did the model succeed to find.
With these defined, F1-score is the harmonic mean of Precision and Recall. Higher the F1-score, better is the model. **F1-score** obtained for the test data on training the model for 300 iterations is **0.91**. Refer Figure 4

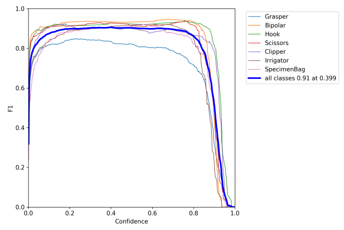


Fig. 4. F1 Curve

C. mAP

1) Confidence: In classification task, we just classify if an image belongs to particular class. But, when we turn the labels soft, we assign a *confidence* for each class which acts as a threshold to decide if an object belongs to that particular class. For a considered confidence c, the model predicts an output with a set of precision and recall. See Fig 5.

mAP is calculated on these woods. AP or Average Precision corresponds to the average of the precision values over the recall values (typically 11 recall values [0.0, 0.1, .., 1.0]. As we also work for bounding boxes, we fix an IoU threshold to calculate the regression loss of bounding boxes and get the best bounding box. mAP of the model is the mean of the Average Precisons for all the classes over a set of IoUs. However, the lateral filter can be compromised and calculate the mAP over a fixed IoU. In our case it is taken as 0.5. We represent such mAP as **mAP@0.5**.

For our task, **mAP@0.5** for 300 iterations is **93.4%**. Refer Figure 5

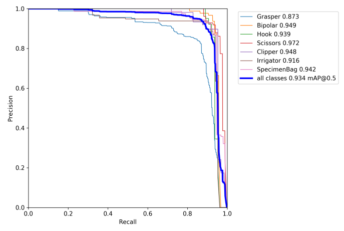


Fig. 5. Precision vs Recall

D. Tool Detection on Visual Data

1) Images: We had also run the trained yolov7 model on batches of size 16 of the test section of m2cai16 tool locations dataset. The result can be viewed at Fig 6.

We can observe there is a very high accuracy in the classification. However the when we compare the last images in the batch, a Clipper is classified as a Hook with 60% probability. This can arise due to the similar visual appearance of the objects' metal part. In the majority of the misclassified cases, the misclassification is due to the presence of only metallic part in the image. However this is a very minor case as the mAP proves it for us.

2) Video: I had taken this video and ran my model on it. The output was visually fine. It can found here.

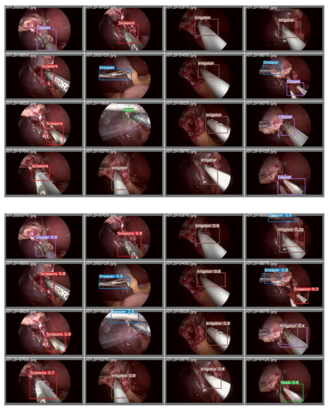


Fig. 6. Tool Detection on a test-batch

E. Github

All the works done for the project can be found in my github repository.

F. Others

We have the graphs of Precision vs Confidence(Fig 7) and Recall vs Confidence(Fig 8).

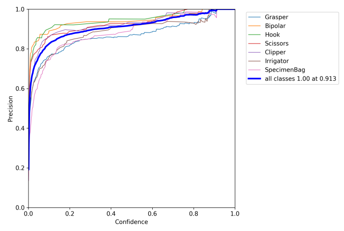


Fig. 7. Precision vs Confidence

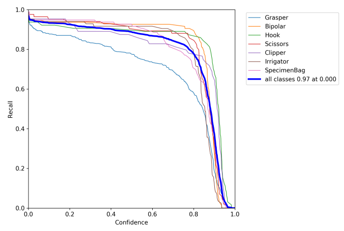


Fig. 8. Recall vs Confidence

References

[1] Dataset
[2] YOLOv1 Paper
[3] YOLOv2 Paper
[4] YOLOv3 Paper
[5] YOLOv7 Paper
[6] Reference Git