TECHNION
Israel Institute
of Technology

# Computer Vision in Surgical Applications (0097222)
# Homework 1

Tomer Sela - 204738470

t.sela@campus.technion.ac.il

Project GitHub:
`https://github.com/Tomer-Sela/Computer-Vision-in-Surgical-Applications-HW1-by-Tomer-Sela`

July 30, 2024

# Contents

# 1 Exploratory Data Analysis

## 1.1 Visualization of Some Images

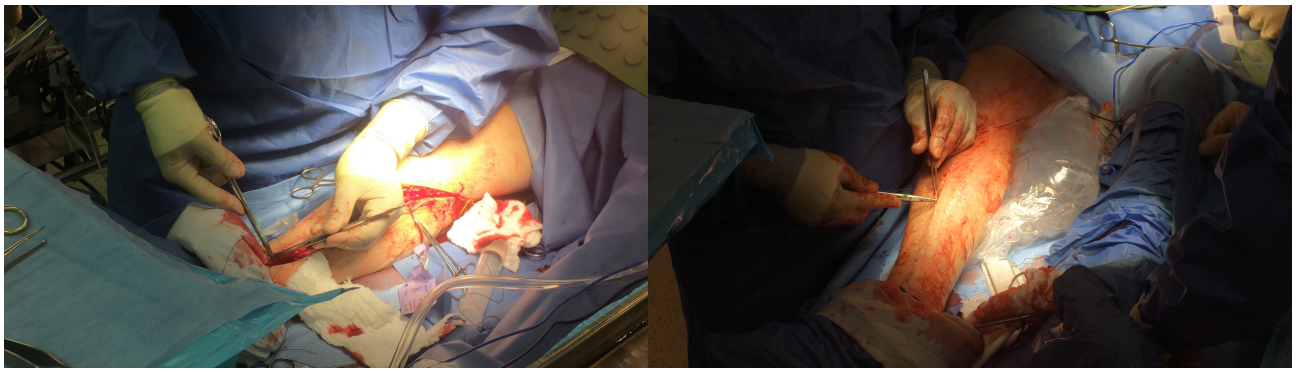


Figure 1: Exemple of two images taken from the already manually labeled data

## 1.2 Insights from looking at the data

Observing the data we can see that we have 3 different data sets:

- **Manually Labeled Images** - Images that were taken out of two videos and have been labeled by hand. The images are very clear and have high resolution. While you can observe from what video each picture was taken from, both are still relatively similar. It is also worth noting that some of the tools we want to recognize are laid on the table, but I do believe it should not be a problem as the model does train on handheld tools and probably will learn the difference easily.
- **Two surgery videos ID** - The Videos are relatively similar, both show a surgery in the same environment with very similar actions in each. The camera does not move the entire video but do change from one to the other as well as a tray with tools in it.
- **Two surgery videos OOD** - The videos are relatively similar to the ID videos but the environment does change a little. Most notably the desk has moved, the lighting is much more dim and the camera angle is slightly changed.

## 1.3 Data distribution analysis

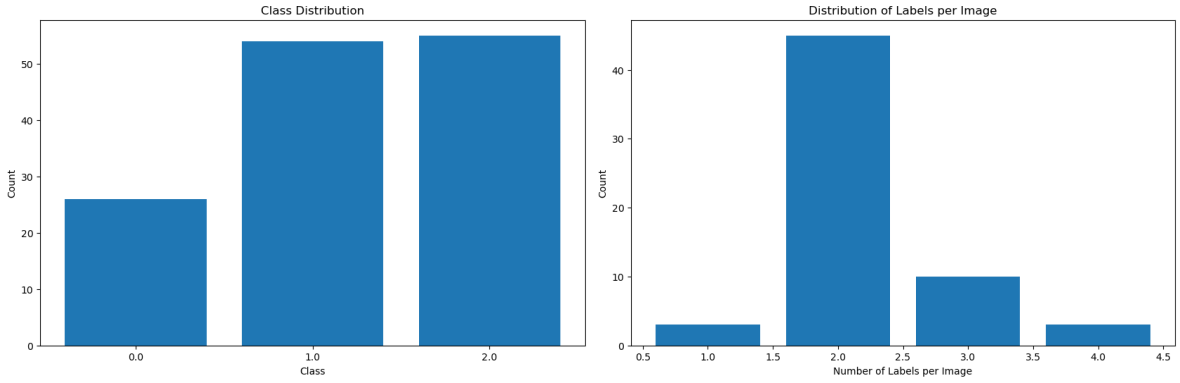



Figure 2: Data distribution analysis of the manual labeled data, 0: Empty 1: Tweezers 2: Needle Driver

From Figure 2a, we understand that both tools have a relatively similar distribution while the empty hand class stands about on about half the number of examples. Figure 2b shows that the majority of pictures contain 2 labels.

# 2 Experiments

In order to train the data I am using the Ultralytics YOLO8v[1] model and interface. The entire training session and data analysis are done by a custom Python code I wrote in Jupyter Notebooks for the experiment and are attached to the GitHub repository. It's important to note that the notebooks require custom changes for each step in the training.

## 2.1 Data loading, pre-processing, and cleaning

All the Data loading and manipulation were done through the custom Jupyter notebook code in order to create a semi-automated process. For data augmentation, I used the YOLO8v default process.
For producing pseudo-labels I use a 3 stages action code. The code first performs a statistical evaluation of the predicted labels to find an appropriate confidence level threshold (in the manner explained in chapter 2.2.2), an example of this stage is shown in figure 3. In the second stage, the code filters the data leaving only labels, and corresponding frames, where **all** classes were predicted above the threshold, thus making sure no half-labeled frame remains. The final stage re-arrange the code for a training section and produce a *data.yaml* file.

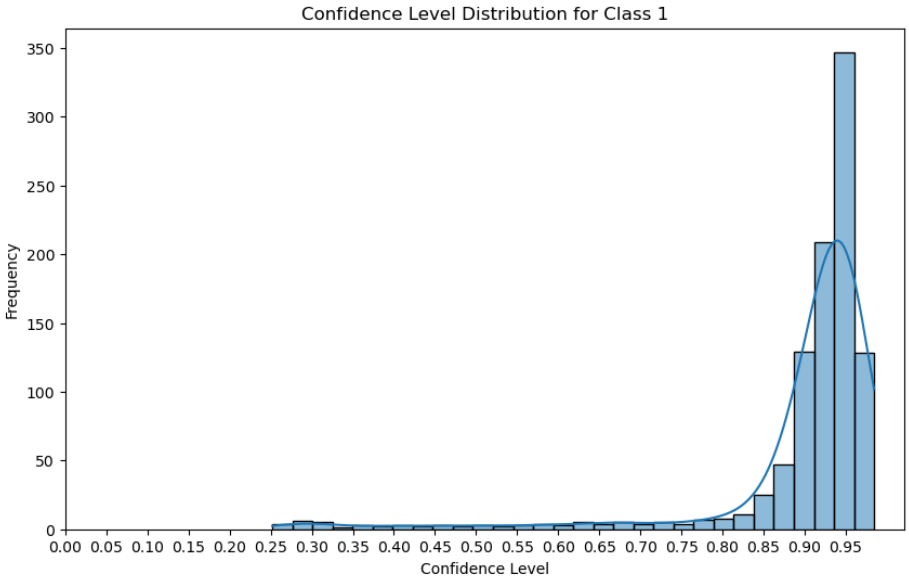



Figure 3: An example of the statistical analysis for the predicted confidence level for the tweezers class

## 2.2 Training Techniques

### 2.2.1 Pre-Training

The Pre-Training was executed on the manually labeled data in order to create The first Model. The training was done with Hyper-Parameters that were found in the cross-validation process and ran for 100 epochs which I find suitable in order to get good and high confidence level initial results, as can be seen in Figure 7.

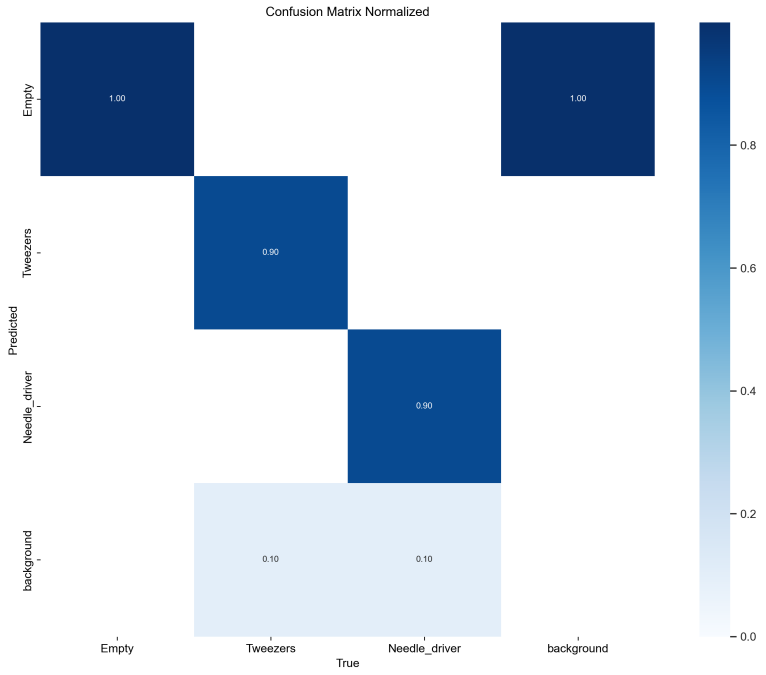


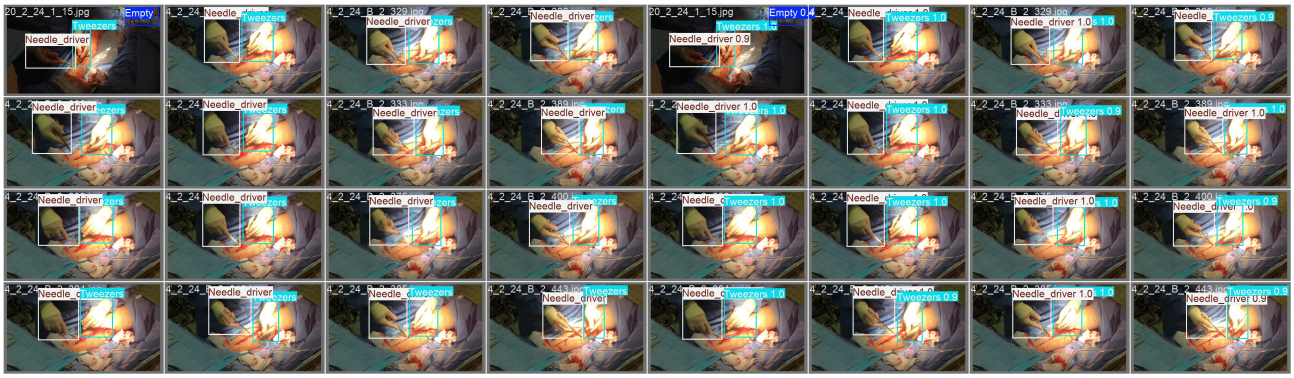

Figure 4: Normalized confusion matrix over the prediction of the In Distribution videos validation set

The validation results, shown in Figure 4 are quite successful, where most of the tools were identified correctly. The model also recognizes an actual empty hand that was not labeled as such giving us the "background as empty" prediction.

### 2.2.2 In Distribution Prediction & Training

I ran the Pre-trained model on both of the ID videos in order to predict pseudo-labels for further training. The model stride was set to 10 frames giving us 3 frames per second in order to get a reasonable number of data that is still meaningful and not too repetitive.
Observing some random examples of the prediction results by hand, I find the model performed very well on the videos, which is expected given that the model was trained over images taken from those videos.

In order to use the prediction as pseudo labels I choose to use the model confidence as a heuristic parameter in order to filter only the most reliable prediction. According to an article by Rafael et al.[2], taking a relatively high confidence level as a filter such that only 10% of the top confidence labels are reminded, should give us the best results. I ran the filter and data processing code described in Chapter with a 0.93 confidence threshold and retrained the model with the same parameters as in the pre-trained. As we train over pseudo-labels, it is hard to assess the model's success and we are left with reviewing some of the validation illustrations by hand to get some sort of a sanity check on the training - which in turn proves quite successful, as the example shown in Figure 6



(a) Validation labels (b) Validation prediction by the model

Figure 5: Validation batch example from the ID validation process

### 2.2.3 Out of Distribution Prediction & Training

While we perform the same procedure for the OOD process as the ID, the results are not as good and early predictions and training attempts performed poorly. The main problem was discovered to be that the model predicted a lot of tweezers as needle drivers to a point where the training forced it to believe that almost every tweeter is a needle driver.
To solve that, I re-ran the prediction stages to produce a normalsizer number of frames (stride set to 3). As we know from chapter 1.3, it is very rare to see the same tool twice in a frame, I ran a custom filter code that clears any frame where the needle driver is seen twice thus cleaning a normalsize number of the mismatch predictions. Afterward, I ran the same cleaning and data arrangement code as in the ID stage which left me with relatively small and more accurate data to train over and thus I redo this process twice.
Again, using an empirical review of the validation batches, I found the model to preform quite well with only a few mismatches in total.

(a) Validation labels (b) Validation prediction by the model

Figure 6: Validation batch example from the OOD validation process

## 2.3 Hyper-Parameter Tuning

In order to find suitable parameters for my training I constructed a Cross-Validation code that ran multiple short training sessions over a number of chosen hyper-parameters out of the Ultralytics YOLOv8 configuration options. The Cross-Validation ran through the *Initial learning rate*, *Final learning rate*, *Weight decay* and *label Smoothing* searching for best accuracy results based on a limited 20 epochs run. Best Parametrs shown in Table 1.

| | Initial learning rate | Final learning rate | Weight decay | Label smoothing |
|---|---|---|---|---|
| Value | 0.01 | 0.001 | 0.0001 | 0.01 |

Table 1: Best Parameters in the Cross-Validation process

### 2.4 Train + Valid Loss Graph & mAP's Graphs

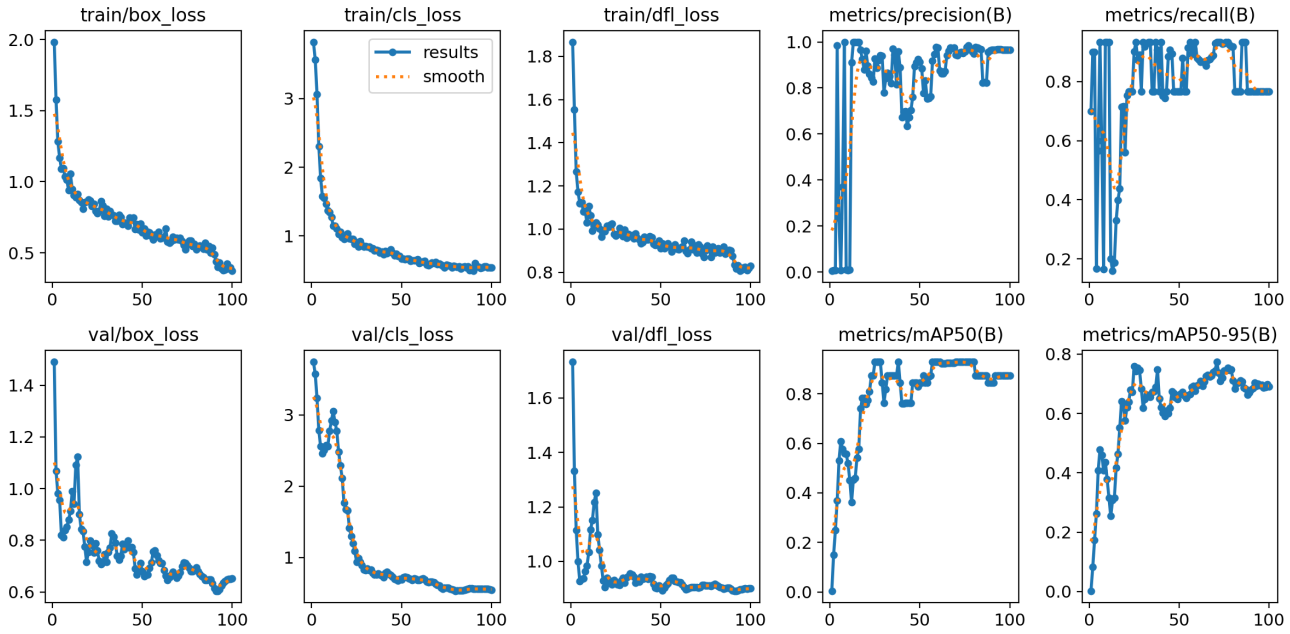



Figure 7: Train + Valid Loss Graph & mAP's Graphs for the first training session over the labeled data

## 3 Discussion and Conclusions

The project has mainly demonstrated the efficacy and ease of use of modern image recognition tools and algorithms and more than that, the surprisingly successful use of pseudo-labeling in order to perform semi-supervised learning. The pseudo-labeling approach proved to be highly effective. Especially when setting a high confidence threshold as a heuristic method in order to refine the prediction for re-training. The second most important thing for me was the pseudo-labeling reviewing and cleaning code, which also took most of my time and effort, but proved crucial in order to get reliable pseudo-labels in order to re-train the model on actual valid prediction and prove once again that a lot of the work in the ML world comes to understand the data and how to handle it in the appropriate way.
The final model performed well on the OOD data, demonstrating its potential applicability in real-world surgical settings. It would be interesting to further test the model on a live camera in a surgical environment to evaluate its performance in real time.

## References

[1] Ultralytics YOLO algorithm `https://docs.ultralytics.com/`

[2] Ferreira REP, Lee YJ, Dórea JRR. Using pseudo-labeling to improve performance of deep neural networks for animal identification. Sci Rep. 2023 Aug 24;13(1):13875. doi: 10.1038/s41598-023-40977-x. PMID: 37620446; PMCID: PMC10449823.