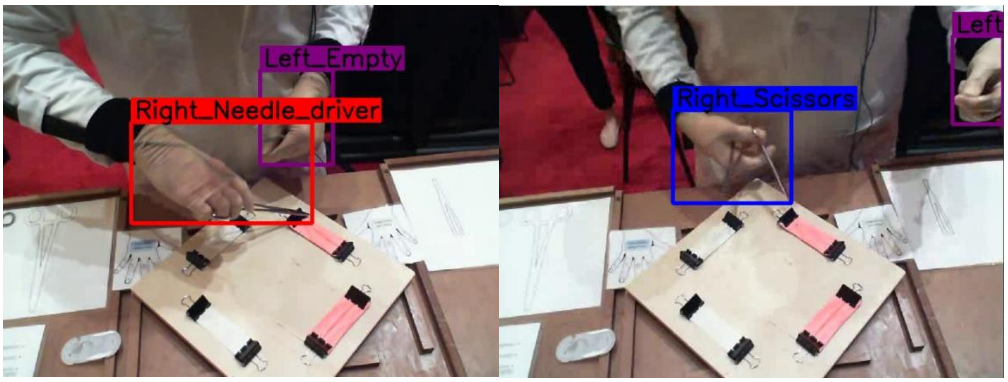
ראיה ממוחשבת בחדר הניתוח דוח תרגיל בית 1

Maxim Matyash, 318828761

David Galambos, 209147297

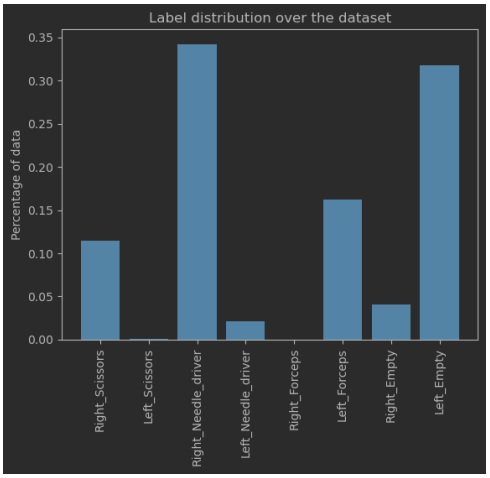
1. Exploratory Data Analysis

a. Visualization of some images



b. Insights from simply "looking" at the data

most of the times the left hand is indeed positioned to the left of the right hand which can cause overfitting. Sometimes a hand can leave the frame – we wont always get a detection.



We observe that the dataset is imbalanced, in particular, the "Right Needle Driver" and "Left Empty" classes dominate the rest of the classes.
The "Left Scissors", "Left Needle Driver" and "Right Empty" are in the extreme minority, accounting for less than 10% of the labels combined, particularly, the "Left Scissors" class has only 3 examples out of 1122.

We theorize that the main factor driving this imbalance is the handedness of the participating surgeons, meaning that the vast majority (if not all) of the participating surgeons were right-handed, which caused the "right" labels and the "left empty"

labels to dominate as they were using their preferred hand.

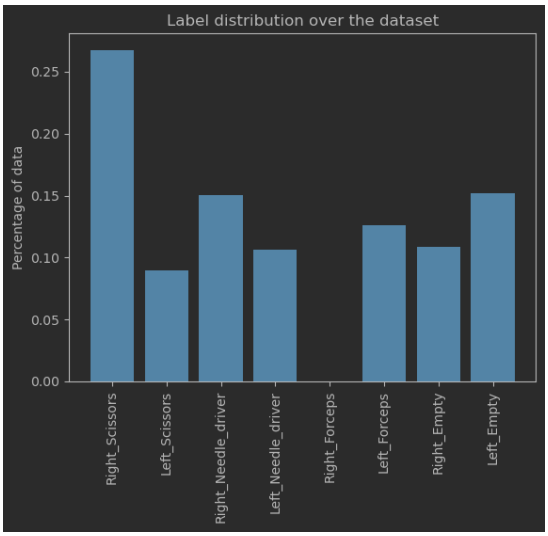
The "Left Forceps" dominates however, due to it being a secondary tool in this procedure and being used in tandem with another tool, therefore it is only used in the left hand since the dominant hand is occupied.

2. Experiments

a. Data loading, pre-processing and cleaning

we ignore images without labels.

we will use augmentation to increase the size of the minority classes, then we will use oversampling and undersampling to balance the dataset.

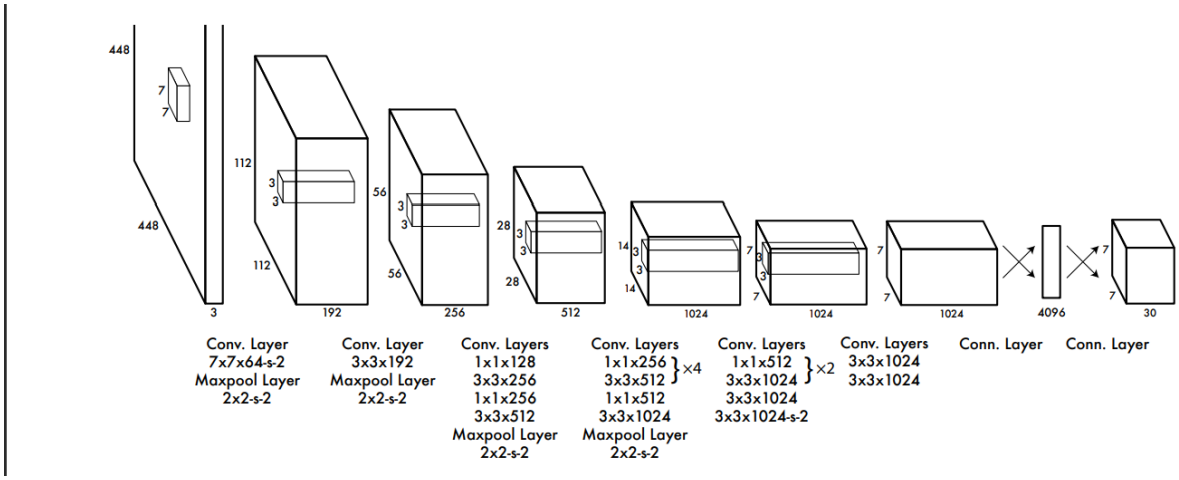


We can see that the data is balanced now and nearly double in size

b. Architecture (describe fully and refer to original paper and code)

and what are your extensions.

Our chosen model was yolov5 which is an improvement over the yolo model1 we've seen in class. As a reminder, yolo is a single stage object detection architecture which looks at the images as a grid of cells and assigns class probabilities to each cell. The original architecture is as such:



While the paper for yolov5 hasn't released yet, we saw that it performs well, and decided to use it. We can see the architecture in Tensorboard:

TensorBoard GRAPHS DISTRIBUTIONS HISTOGRAMS TIME SERIES

Search nodes. Regexes supported.

Fit to Screen

Download PNG

Run (1) exp4

Tag (2) Default

Upload Choose File

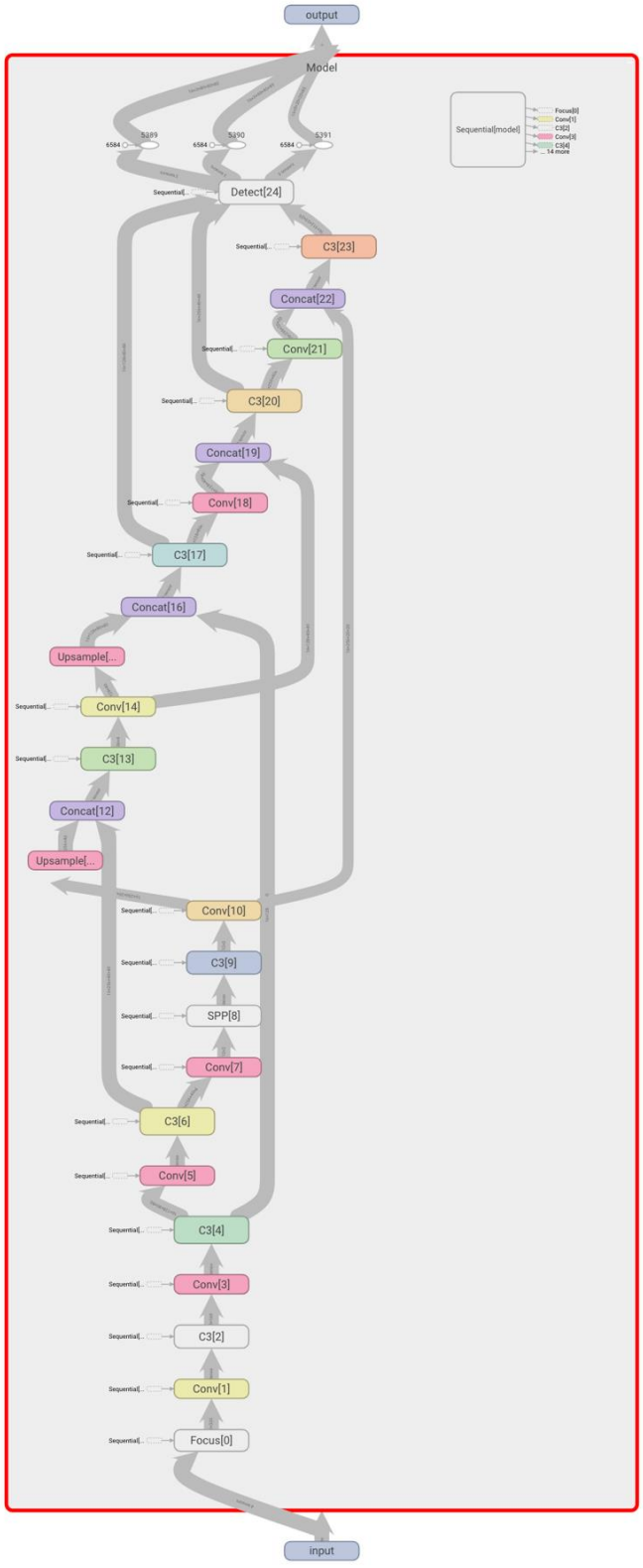
- (•) Graph
- ( ) Conceptual Graph
- ( ) Profile

Trace inputs

Color
- (•) Structure
- ( ) Device
- ( ) XLA Cluster
- ( ) Compute time
- ( ) Memory
- ( ) TPU Compatibility

colors same substructure

unique substructure



Close legend.

Graph (* = expandable)

- Namespace* ?
- OpNode ?
- Unconnected series* ?
- Connected series* ?
- Constant ?
- Summary ?
- Dataflow edge ?
- Control dependency edge ?
- Reference edge ?

The main new addition we noticed is the C3 module, which is called Concentrated-Comprehensive Convolution. In essence, it's an improved dilated convolution which reduces the number of parameters by half.2

Link to the yolov5 github: https://github.com/ultralytics/yolov5

YOLO paper: https://arxiv.org/abs/1506.02640

C3 Paper: https://arxiv.org/abs/1812.04920

c. Loss Functions, explain the role of each term of the loss function

d. Optimizers

e. Regularization

f. Hyper parameter tuning

most of the above was done by the implementation we used: adaptive learning rate that decreases over epochs, weights decay and early stopping. We tested two optimizers – adamw and sgd and saw that adamw got slightly better results

For every experiment, show:

See notebook for model results, as well as the graphs under yolov5_ws/yolov5/runs.

In general, we saw good performance on train and validation sets, which decreased majorly on the test set, mostly due to the extreme minority labels which were less than 1% of the data, but has horrible mAP values and decreased the overall mAP by a lot.

3. Tool usage evaluation on test set videos:

For the videos we loaded the model using torch and opened the video with CV2.
We save the last 45 predictions for each hand and on each frame we run our model to get the raw predictions.
the raw predictions are displayed with the boxes and replace the oldest prediction in the 45 predictions list
the top label reported as the segmentation is the most appearing label in the last 45 frames.

To prevent miss match of segments with the ground truth, we make sure that each hand gets exactly one prediction per frame.

The average performance on the 5 videos:

weighted precision, 0.949048
weighted recall, 0.945629
f1_micro, 0.945629
f1_macro, 0.760445
accuracy, 0.945629

4. Discussion and Conclusions

We have witnessed a good case of domain adaptation with the fine tuning of yolo5 for our data. There were more challenges discovered on the way then we expected, frames with more or less then 2 predictions, tinkering with configurations of the model and balancing the data set.

There is overfitting to patterns in the data like most used tools, similarity between tools, position of tools and hands. The more “expert knowledge” we employed the better were the results, forcing the amount of predictions, smoothing and balancing.

We tried using only 15 frames for smoothing but its apparently a too short time and the segments weren’t stable at all (around 100 per video). On the other hand, using 45 frames did result in more stable segments (around 10 per video) but it takes a long time to pass to the new correct label. More sophisticated smoothing could have improved the segmentation.