Project Lab Human Activity Understanding
SoSe 2023

Project Jarvis

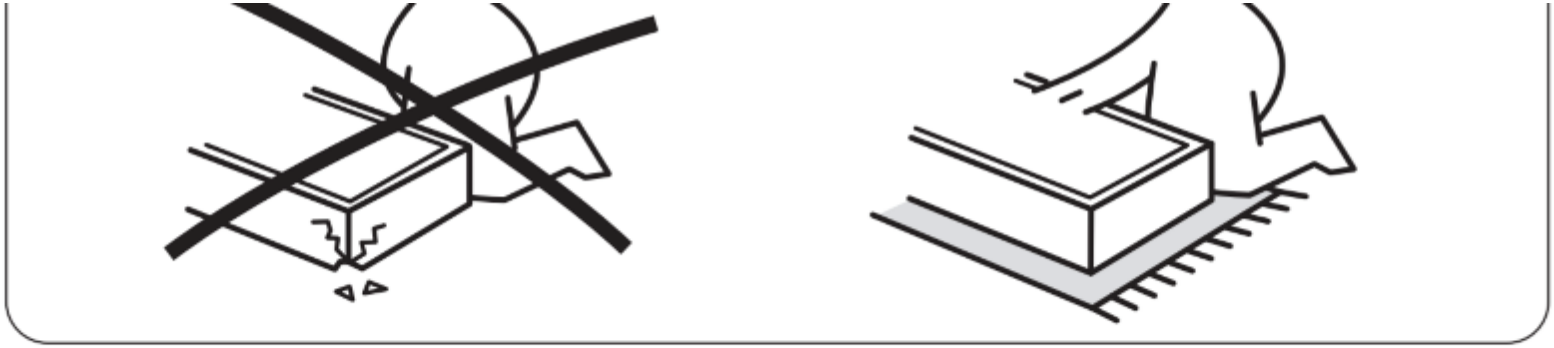
Assembly Tool

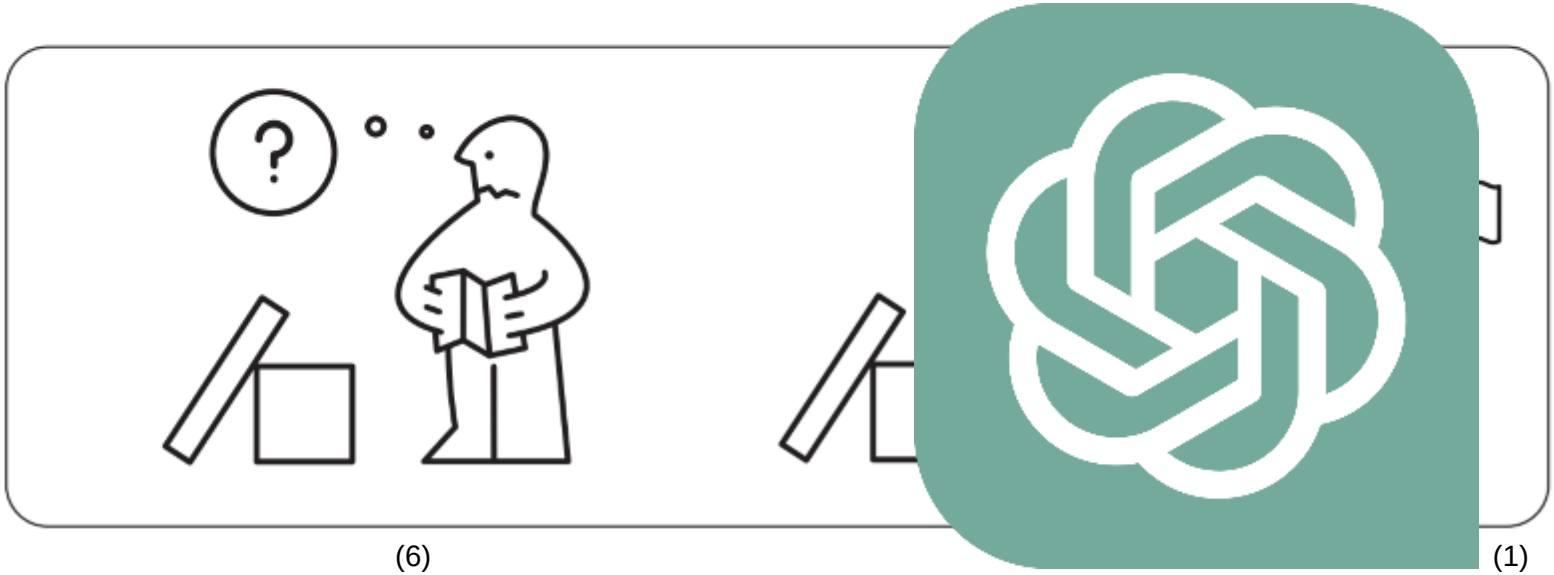
Group C: Zain Amir, Marc Benesch, Hamas Khan, Mohammad Asif Ibna Mustafa

Contents

- Motivation & Purpose
- High Level Overview
- Pipeline:
  - Object Detection
  - Hand Landmarks
  - MLP and LSTM
  - Transformer Network
- LLM Integration
- Visualization & Demo

Motivation & Purpose





(6) (1)

Motivation & Purpose

- Central question:
  Is it possible to use a Large Language Model (LLM) to replace laborious and long instruction manuals?
- Possible use cases:
  - Digital Assistant via Apple Vision Pro (or similar)
  - FAQ-Bot with image-to-text capabilities to help customers in need
  - feel a bit like Iron Man

(2)

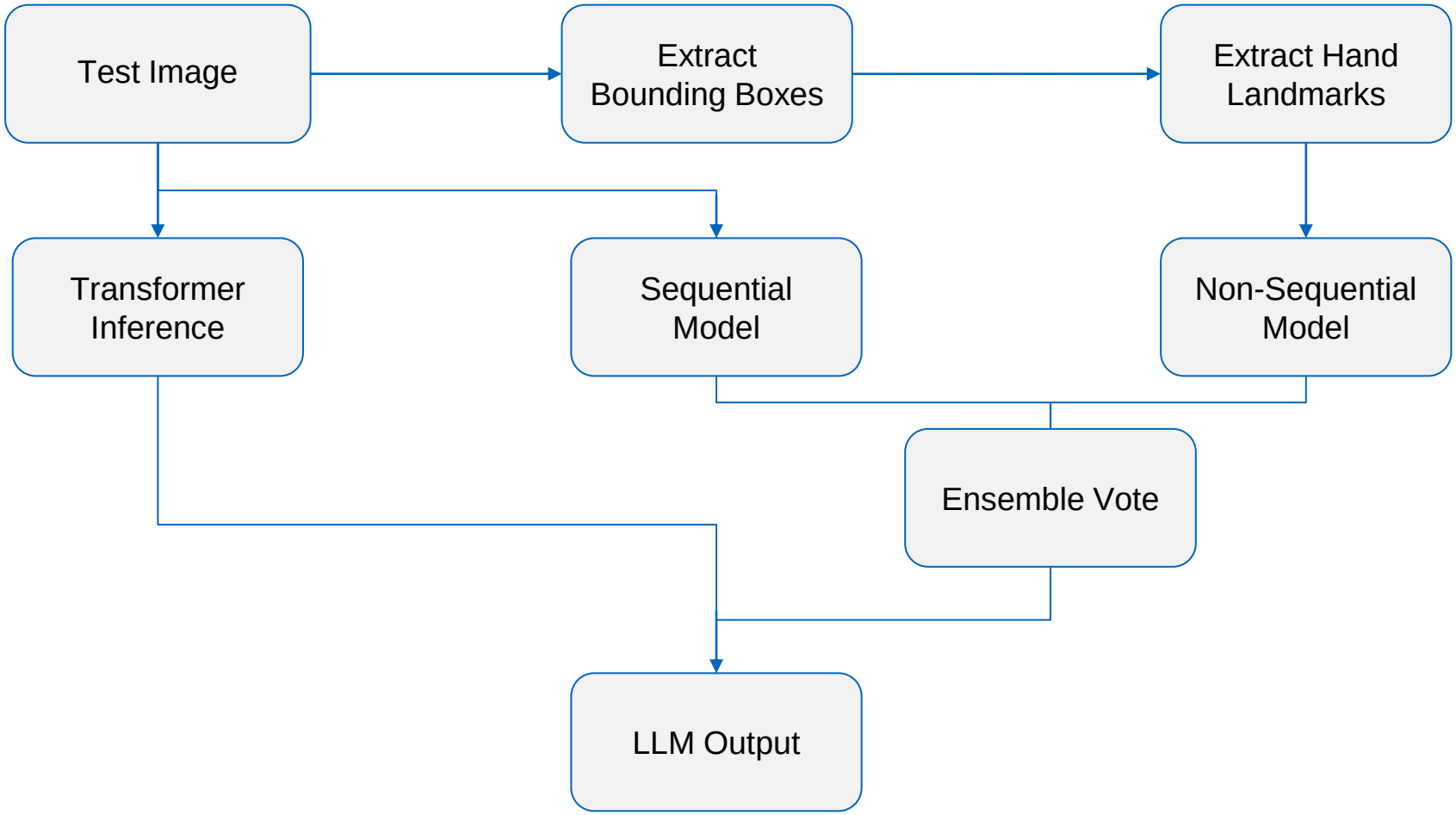
Motivation & Purpose

- Go even further?

(3)

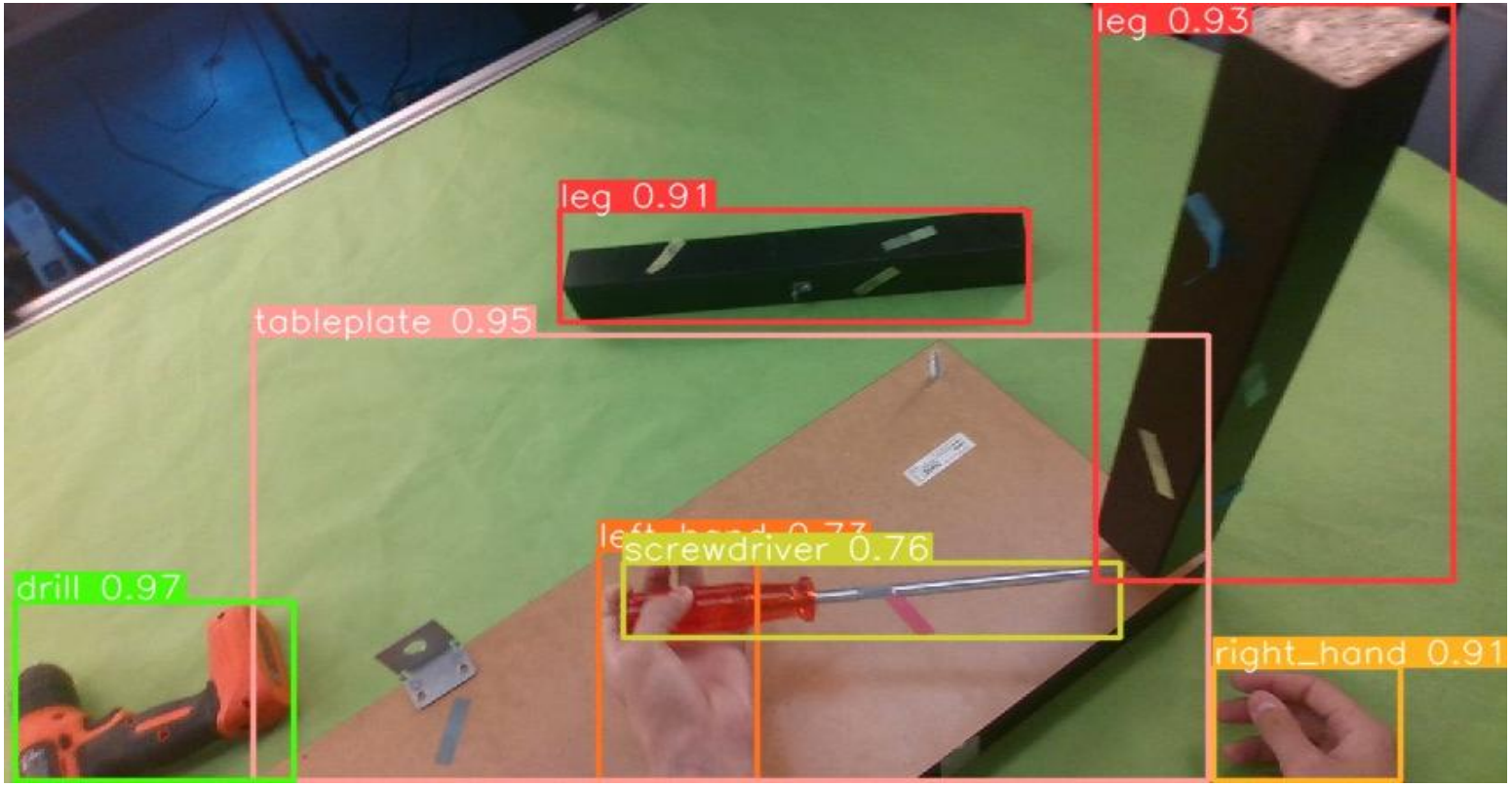
- Based on input: show the next instruction on VR interface (disclaimer: we did not implement this)

Program Flow

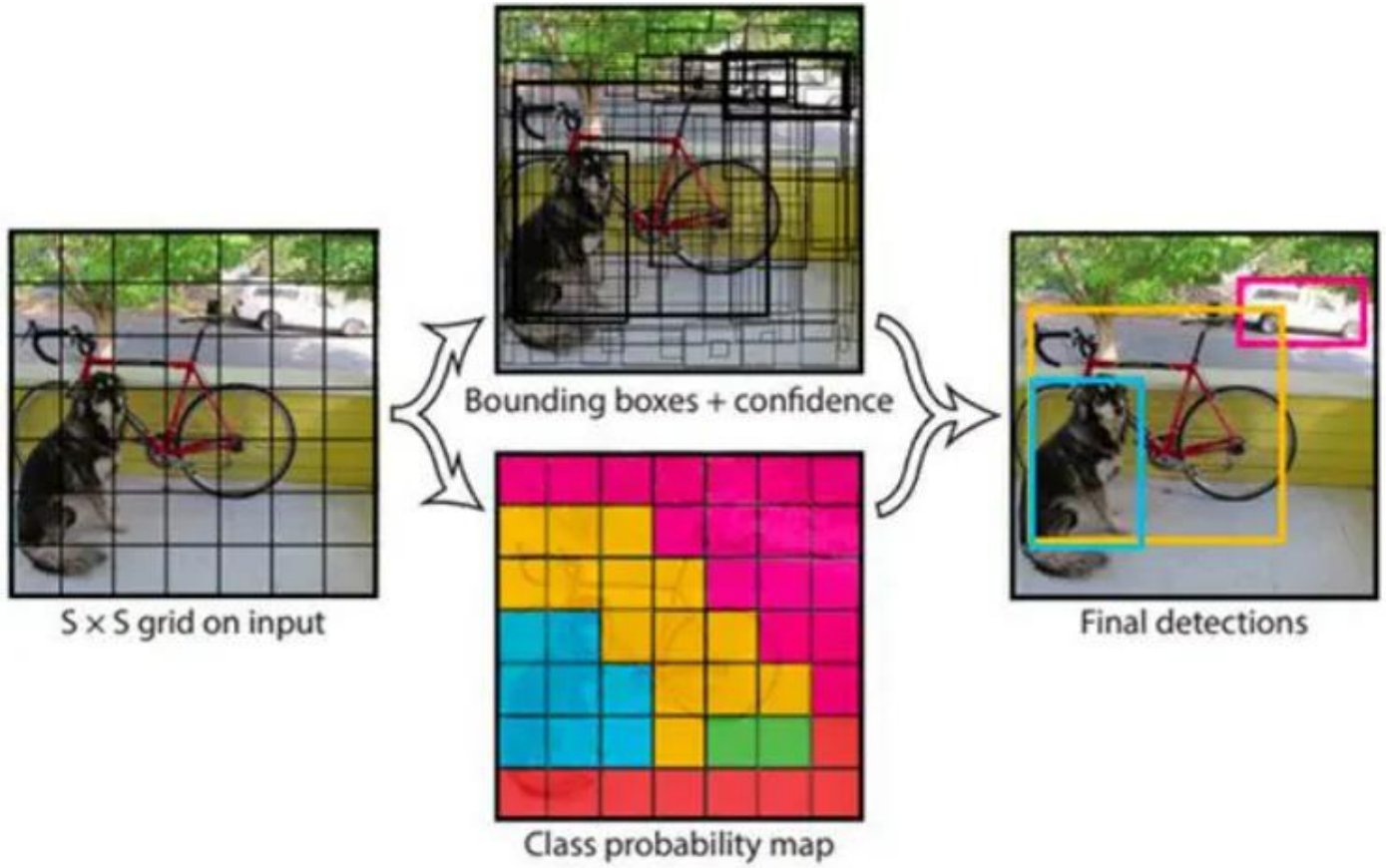


Object Detection - Introduction

Object detection is a computer vision task that involves identifying and locating objects in an image or video.

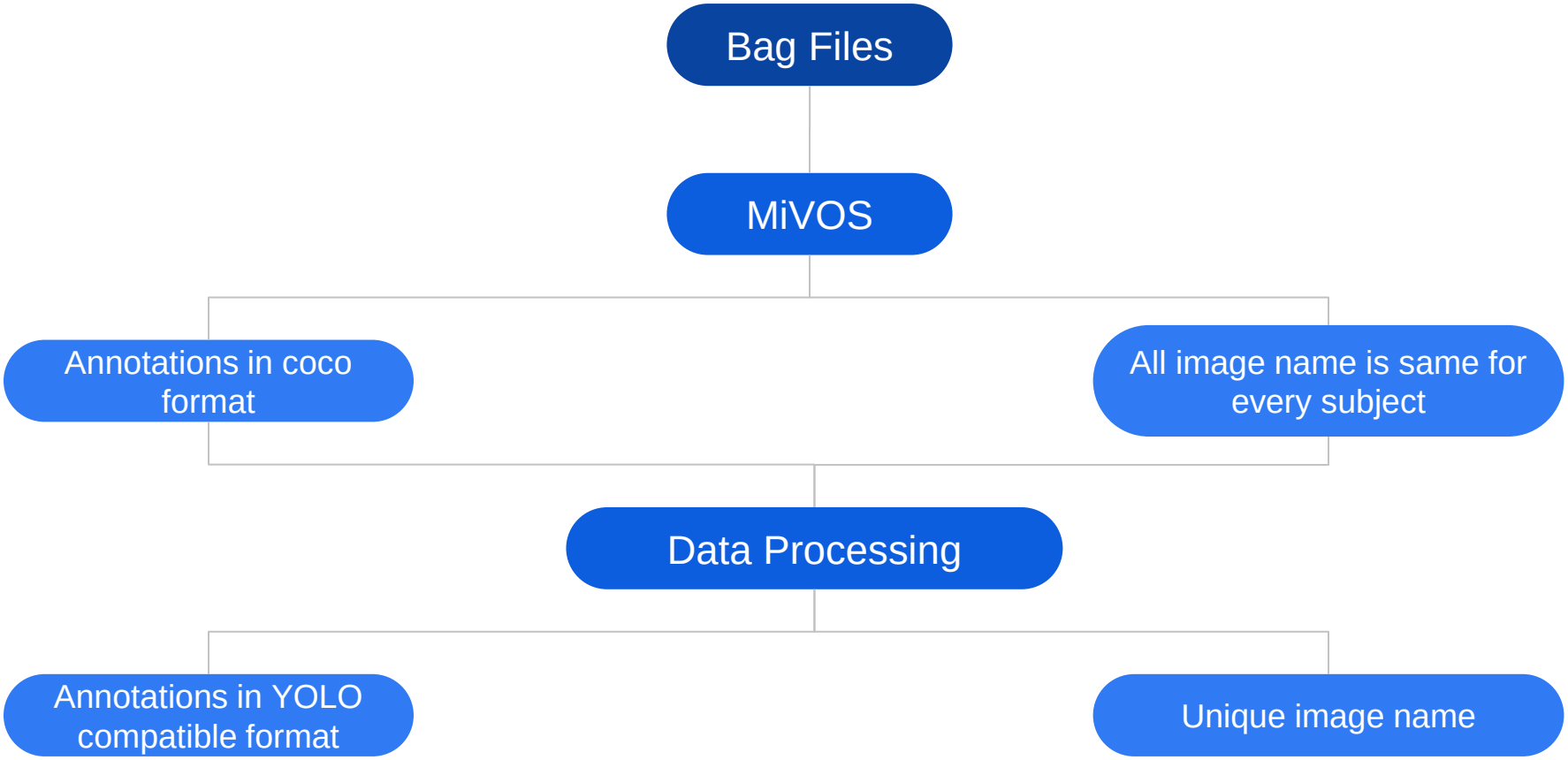


YOLO - Introduction



YOLO - Introduction

Data Processing for Object Detection



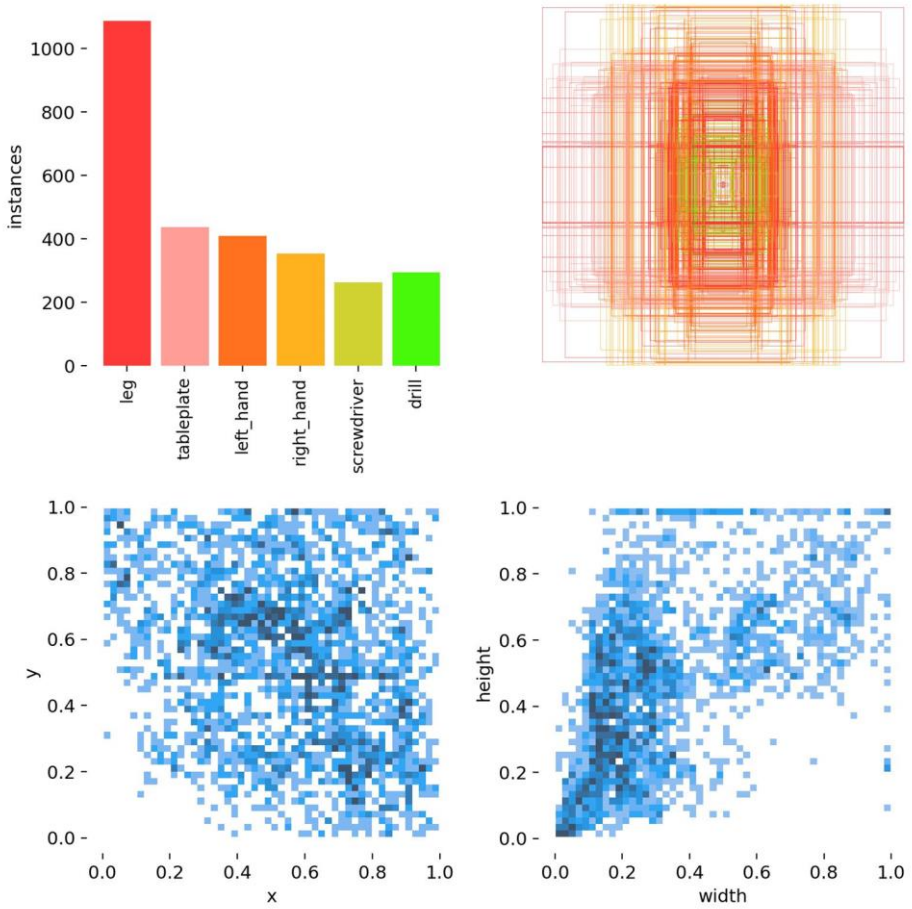
Annotation

Annotations:

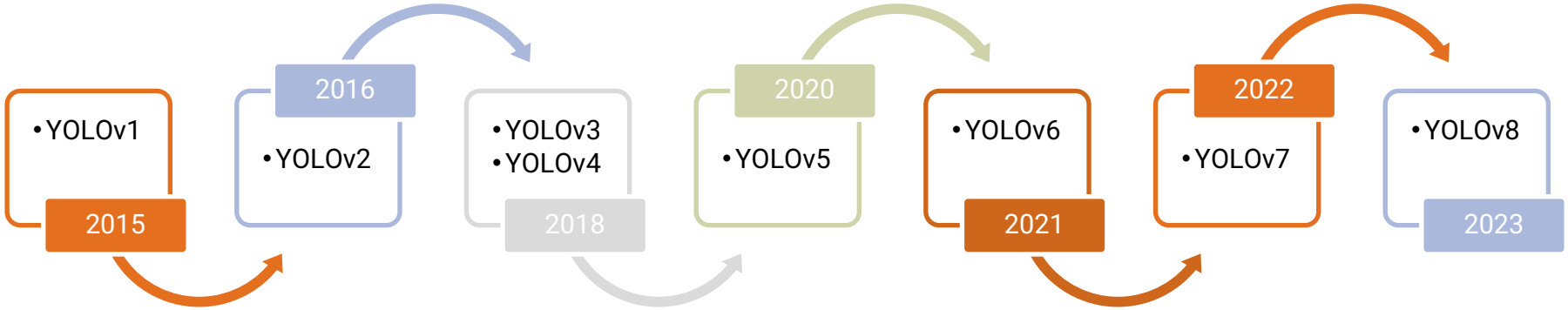
- Annotated 6 bag files
- Ego Perspective

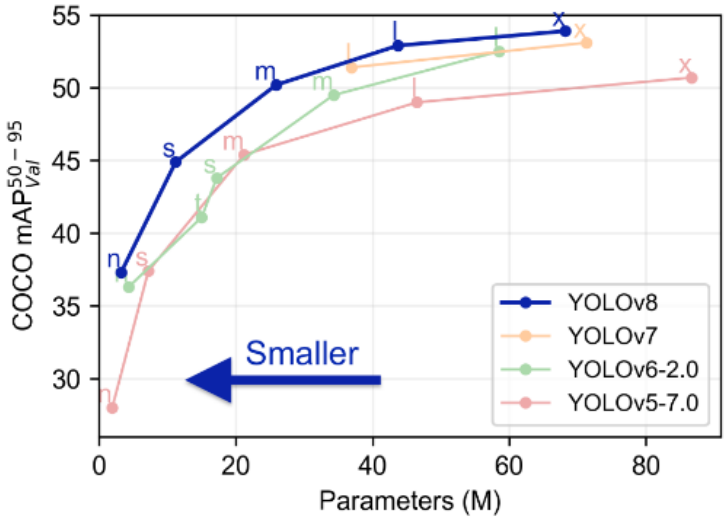
Annotated classes:

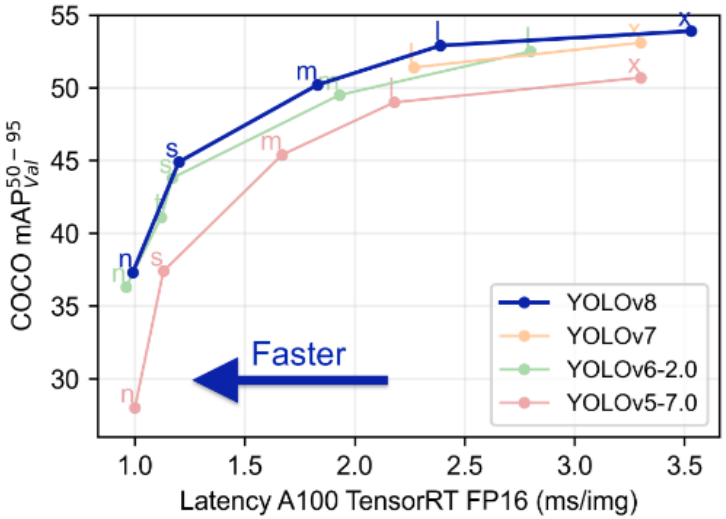
- Legs (4 instances)
- Tableplate
- Left_hand
- Right_hand
- Screwdriver
- Drill



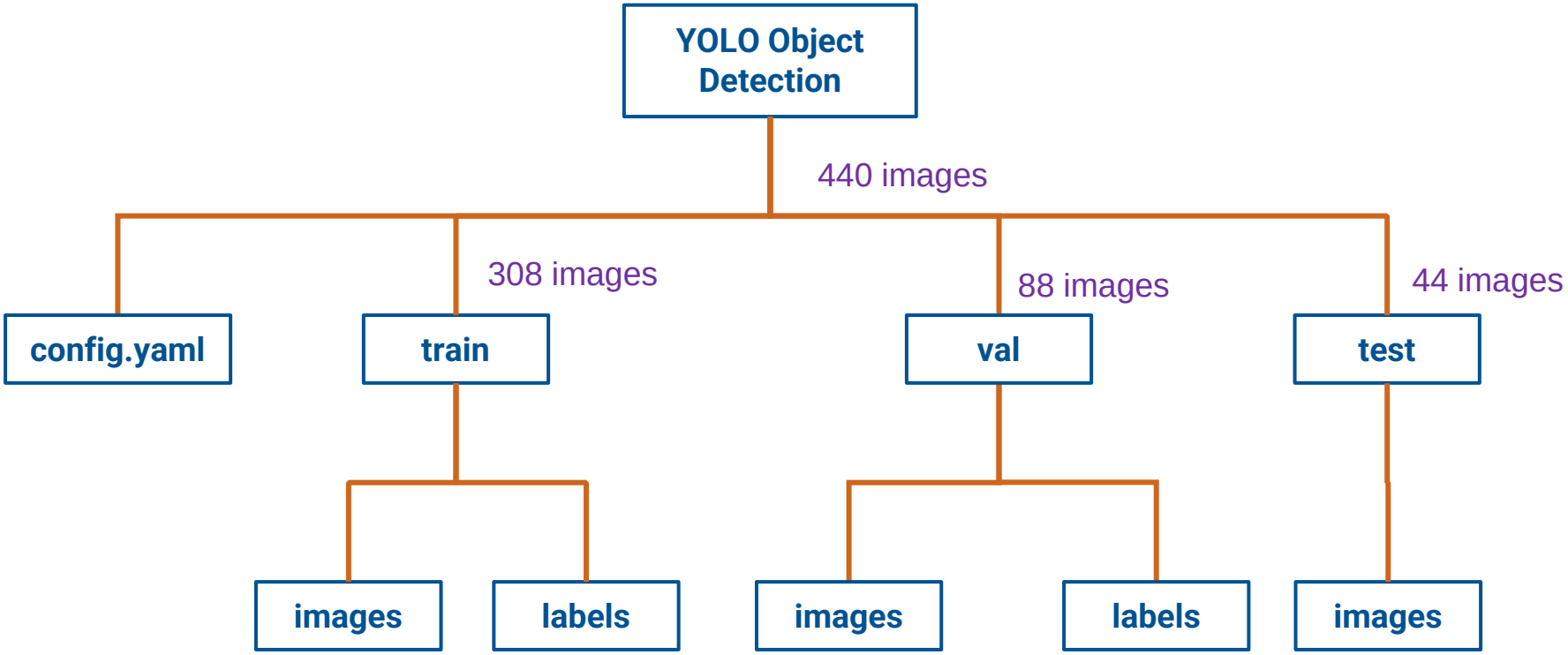
YOLOv8 – Why?


•YOLOv1
2015
2016
•YOLOv2
•YOLOv3
•YOLOv4
2018
2020
•YOLOv5
•YOLOv6
2021
2022
•YOLOv7
•YOLOv8
2023


COCO mAP$^{50-95}_{val}$
Parameters (M)
Smaller
YOLOv8
YOLOv7
YOLOv6-2.0
YOLOv5-7.0

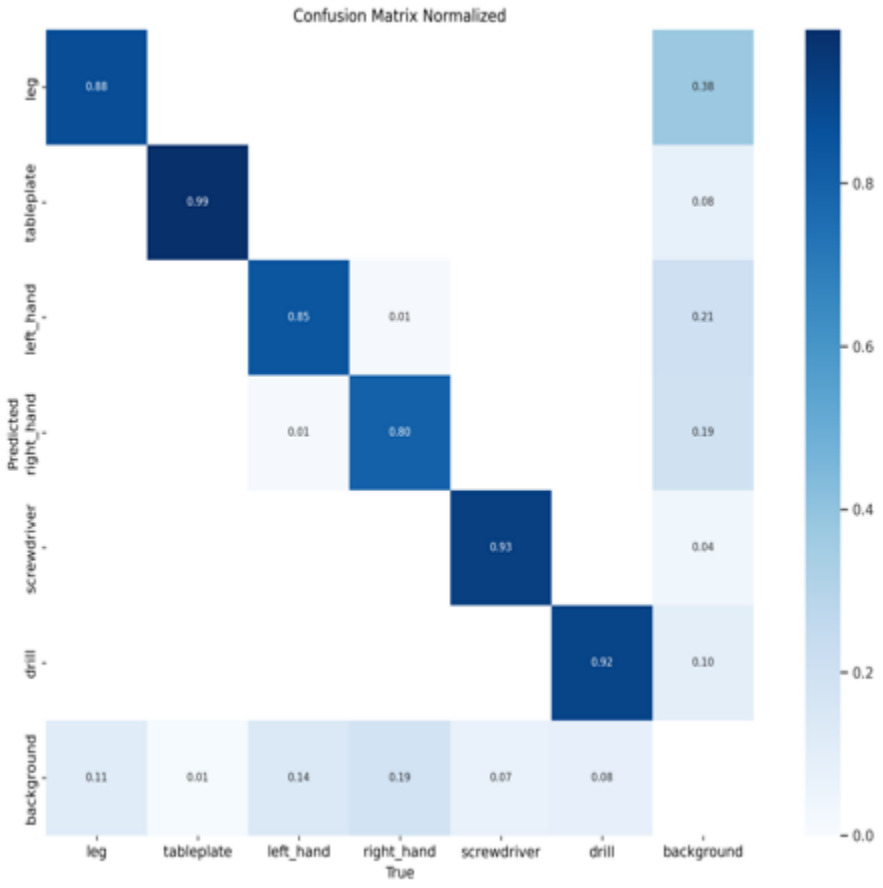

COCO mAP$^{50-95}_{val}$
Latency A100 TensorRT FP16 (ms/img)
Faster
YOLOv8
YOLOv7
YOLOv6-2.0
YOLOv5-7.0

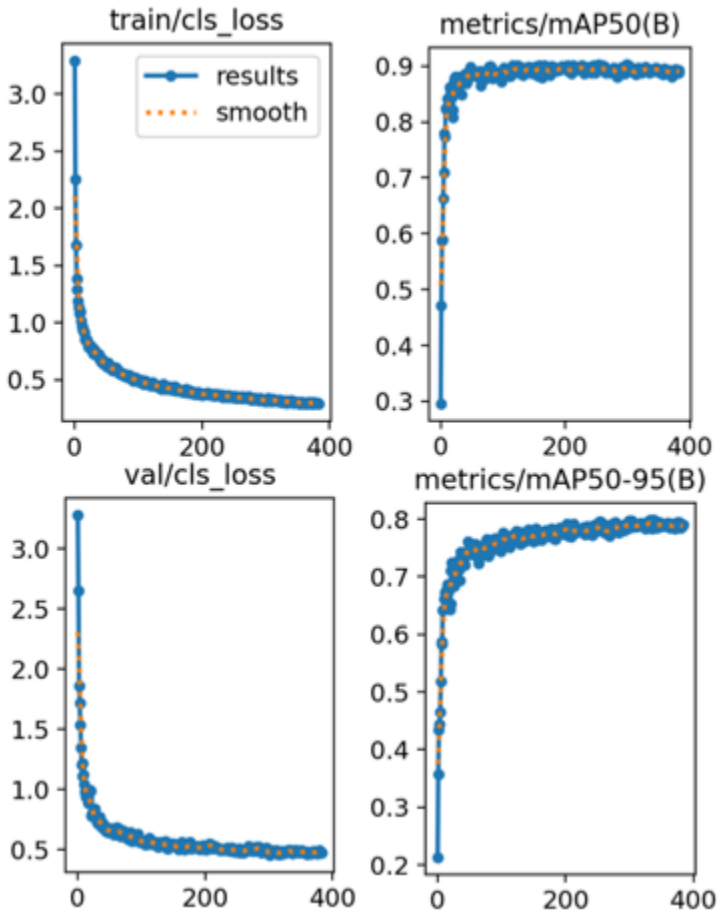
YOLOv8 - Preparation



YOLOv8(n)

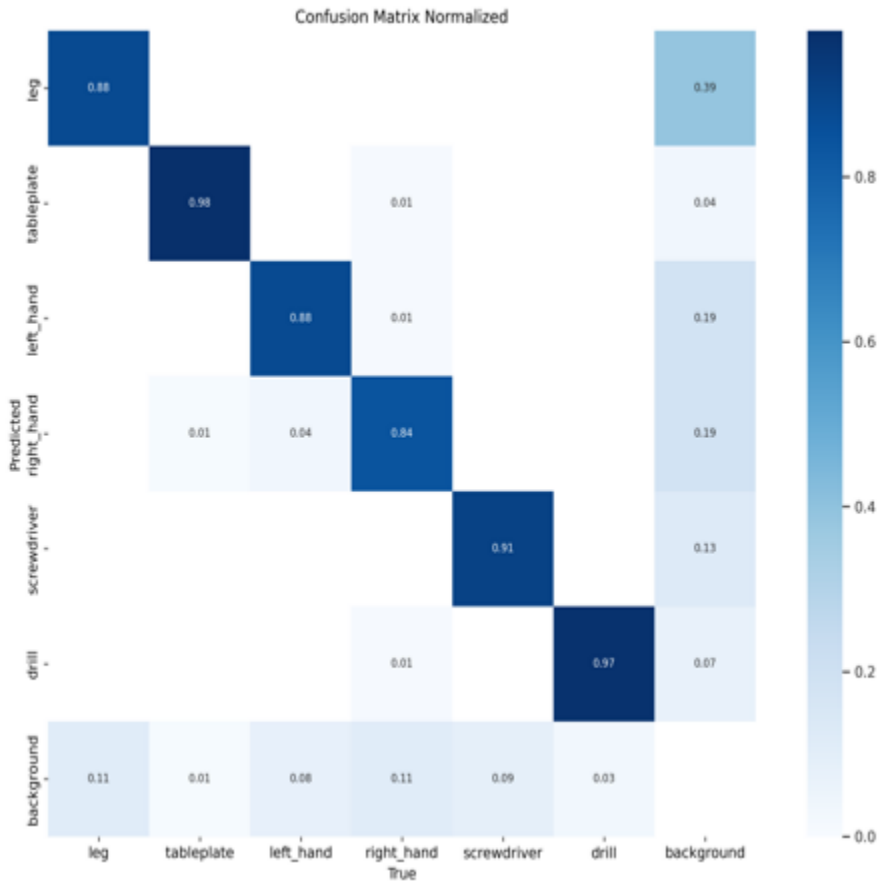
@mAP50: 0.893; @mAP50-95: 0.798

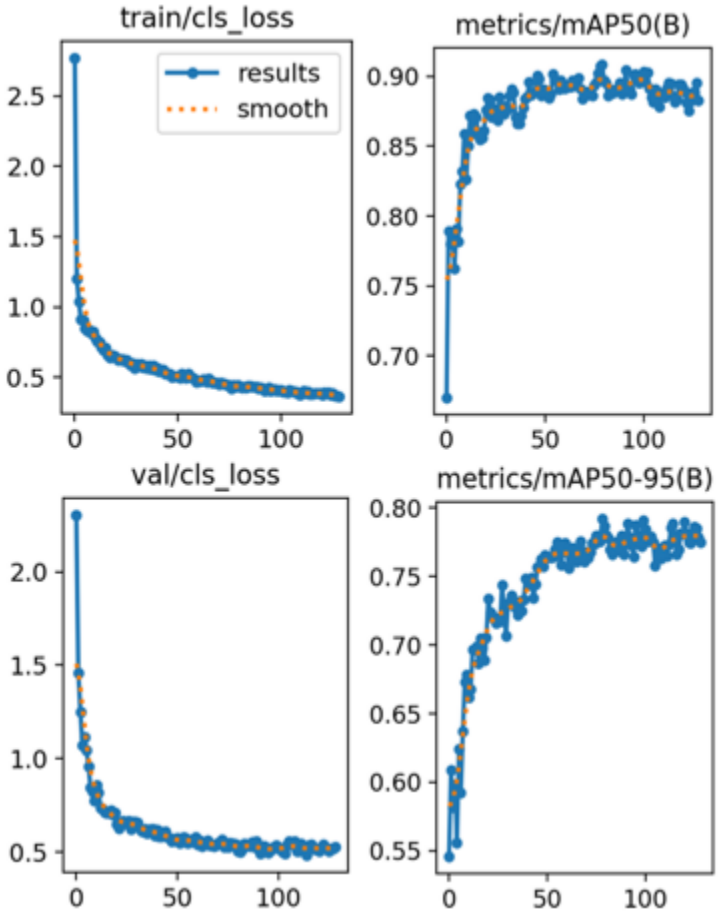




YOLOv8(s)

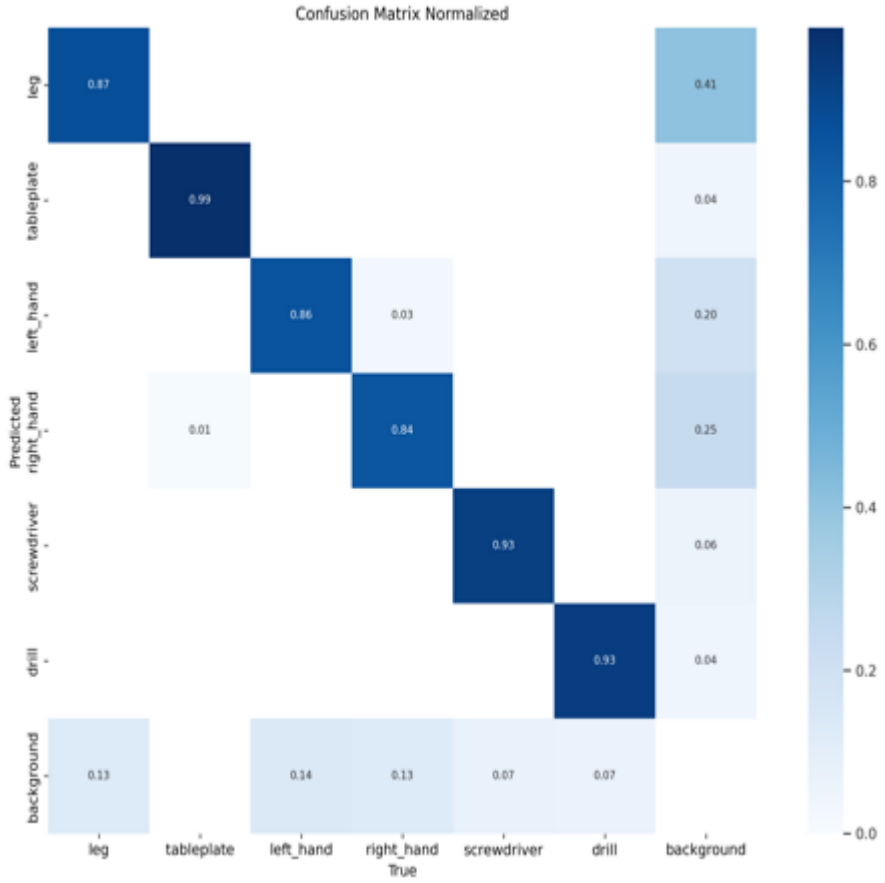
@mAP50: 0.907; @mAP50-95: 0.791

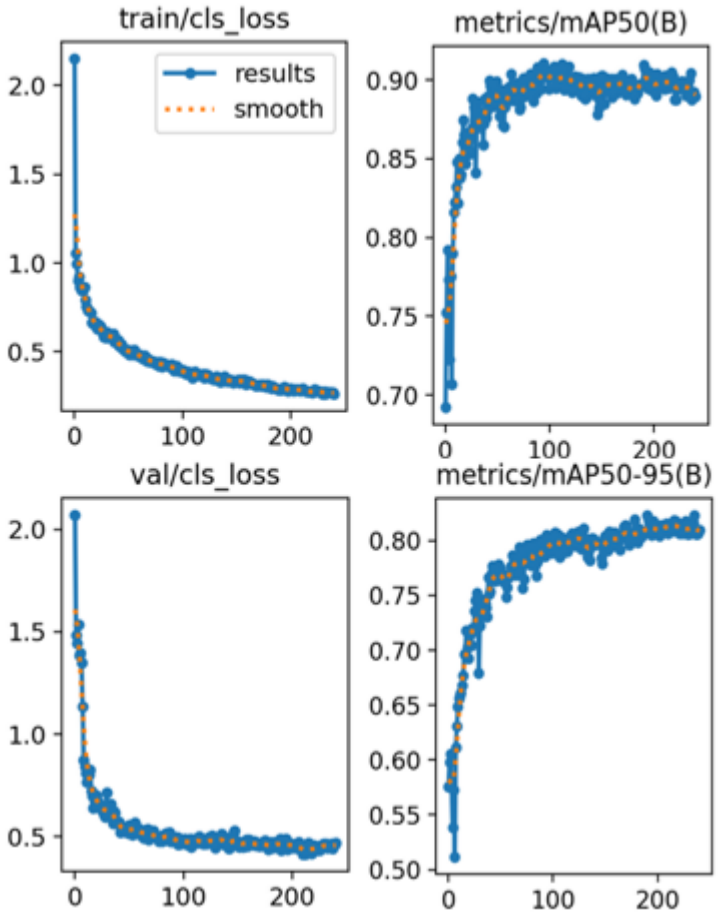




YOLOv8(m)

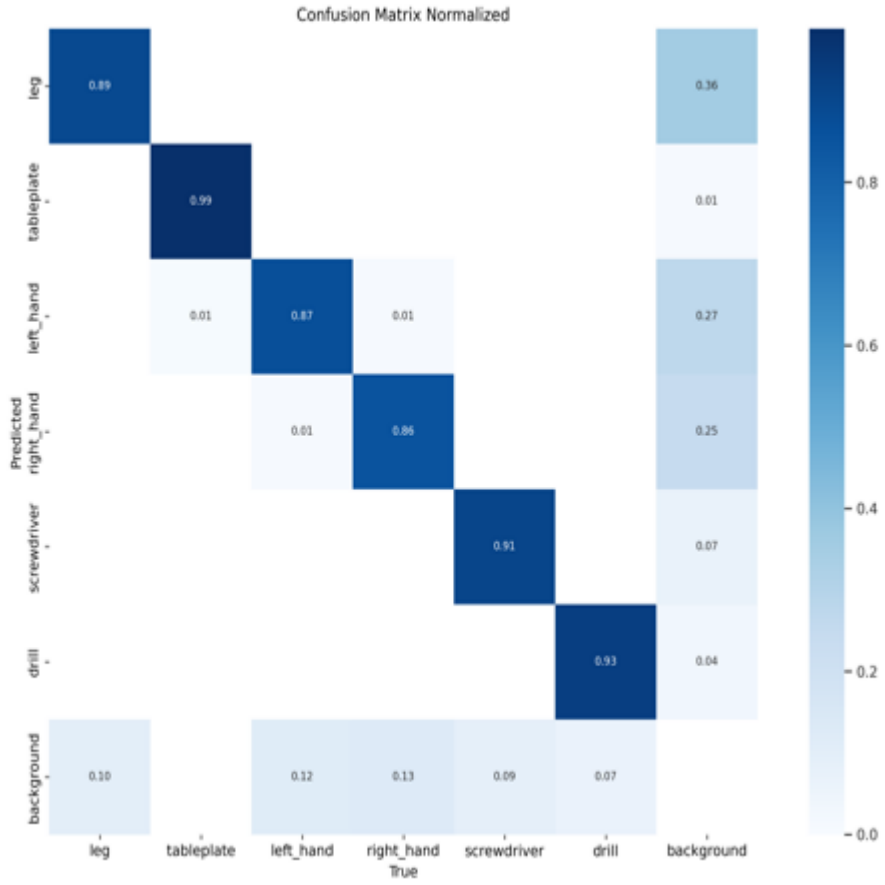
@mAP50: 0.909; @mAP50-95: 0.824

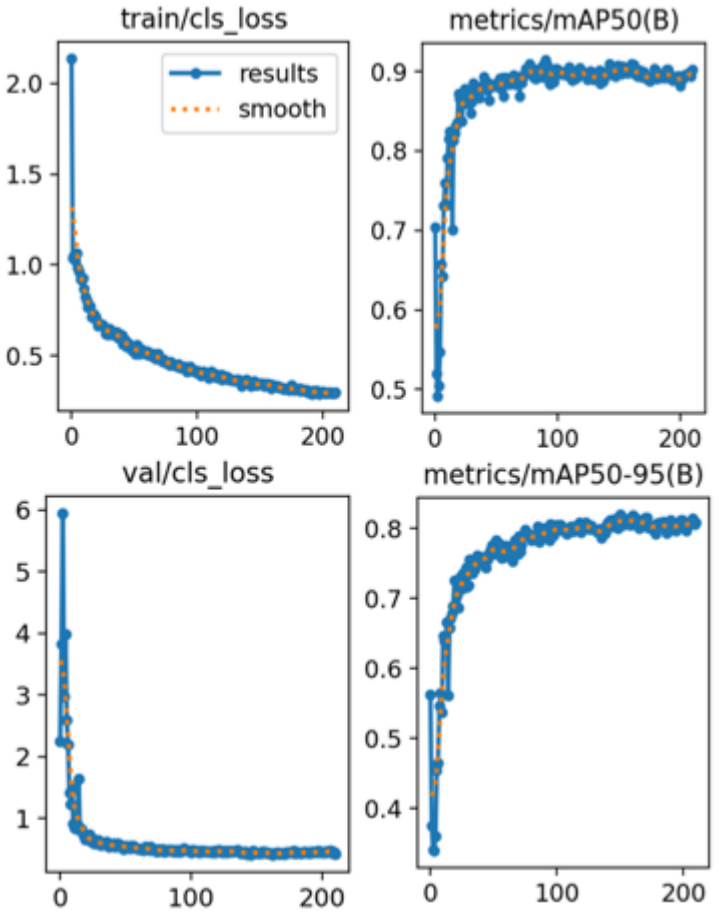




YOLOv8(l)

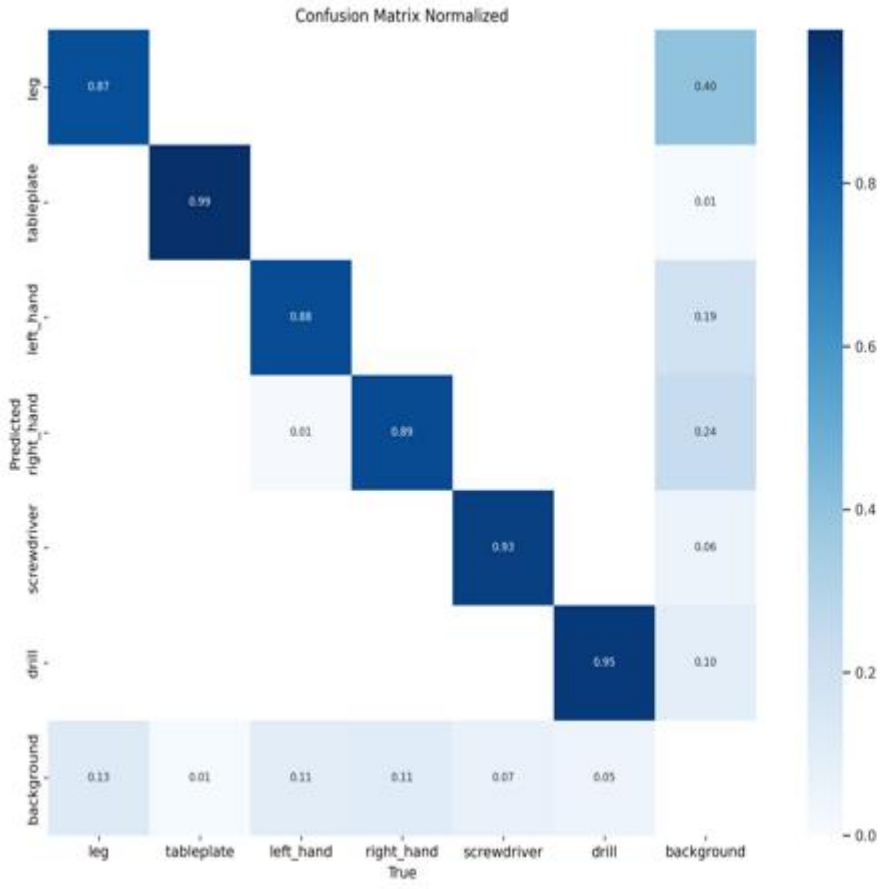
@mAP50: 0.910; @mAP50-95: 0.820

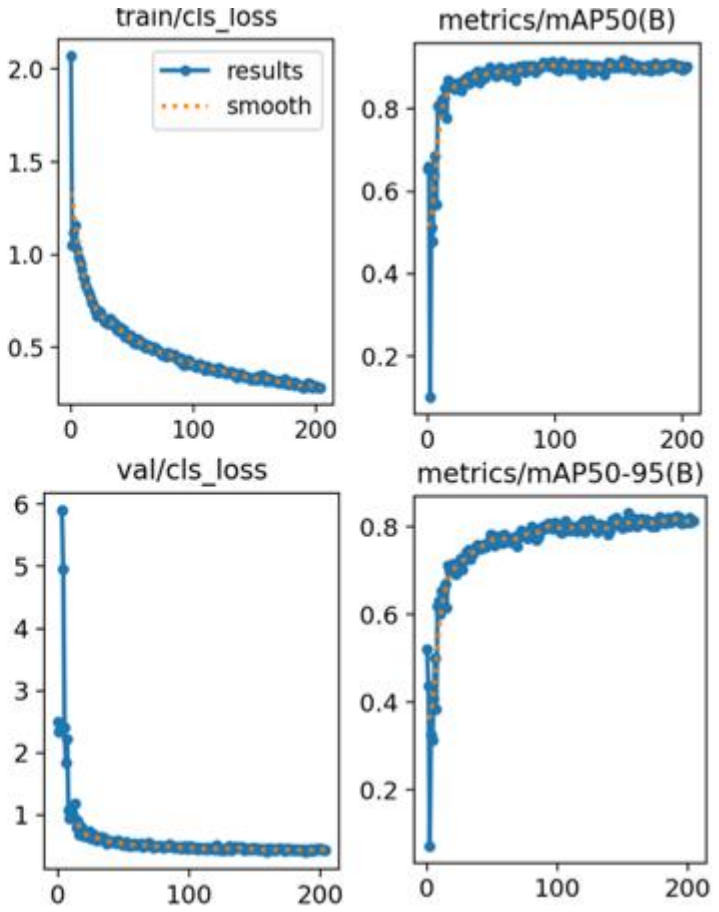




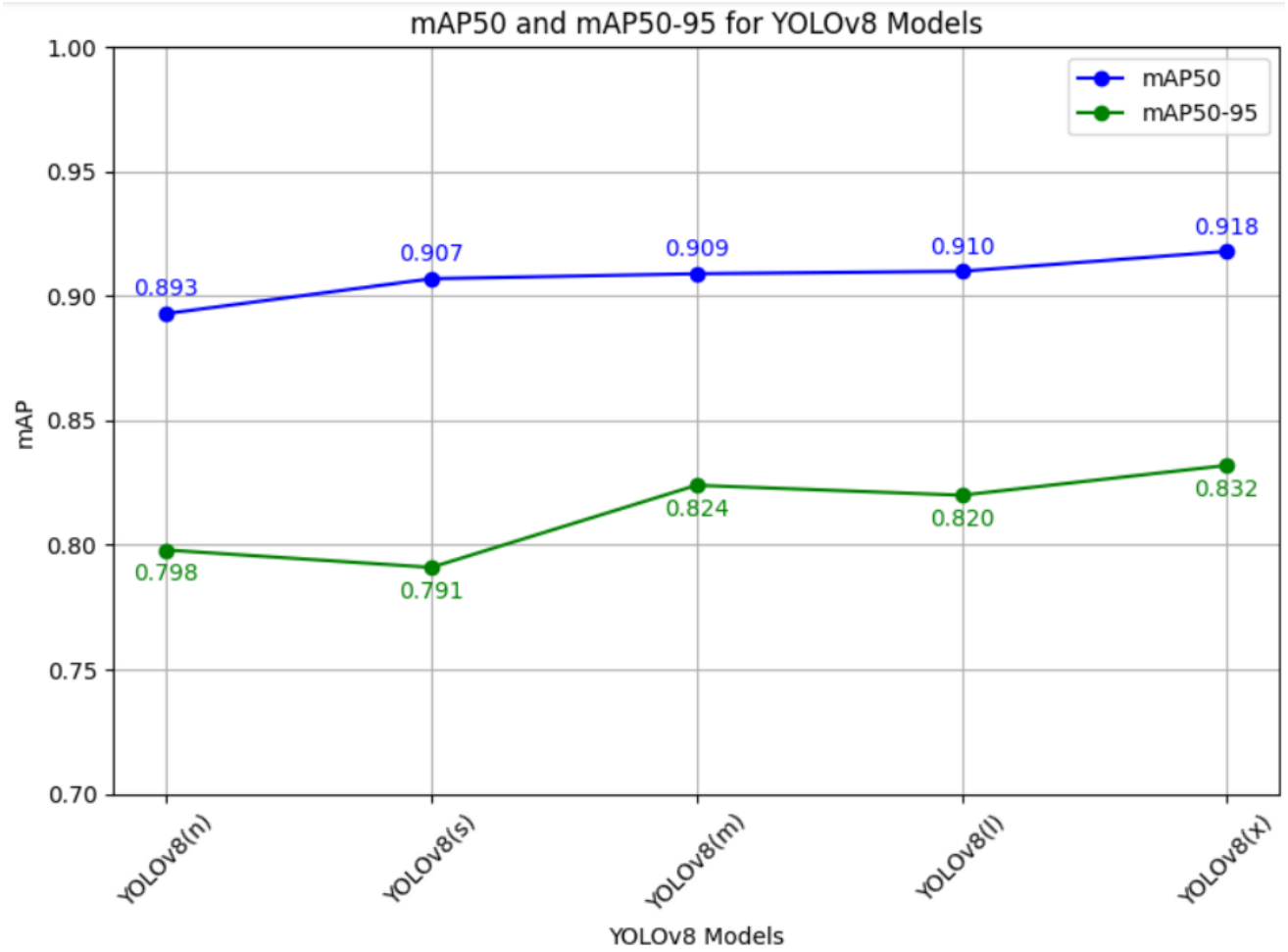
YOLOv8(x)

@mAP50: 0.918; @mAP50-95: 0.832

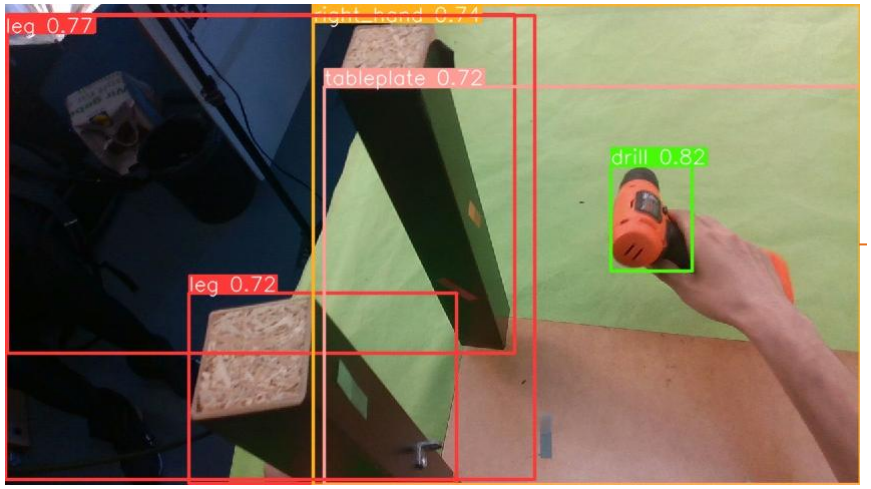




YOLOv8 – Which Model?

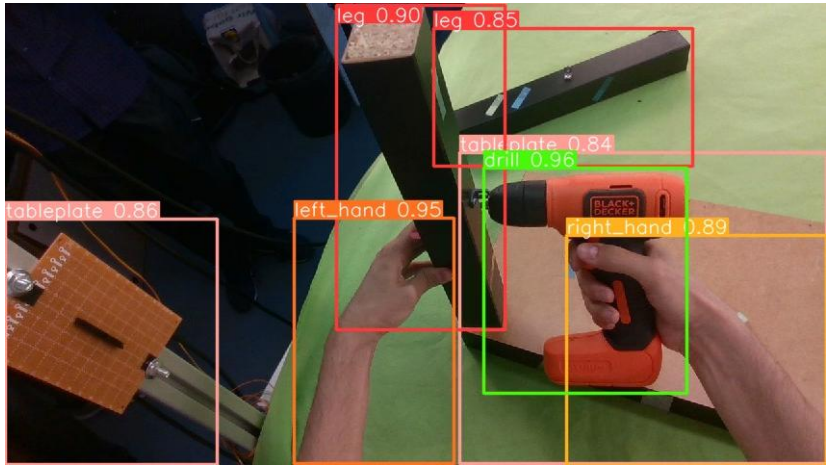


YOLOv8 - Challenges

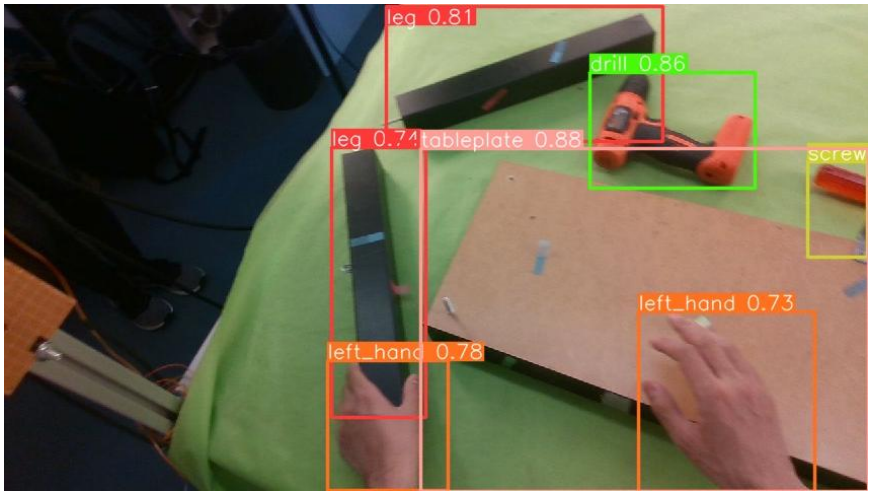


#1:
Background is detected as leg

#2
Circuit board is detected as tableplate

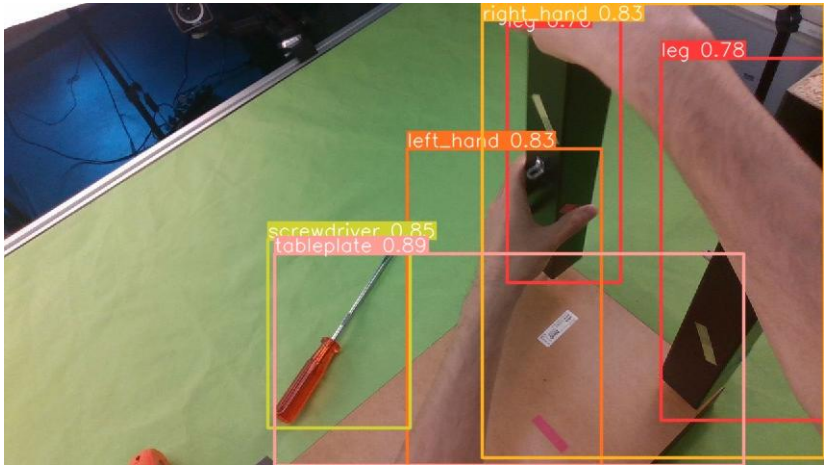


YOLOv8 - Challenges

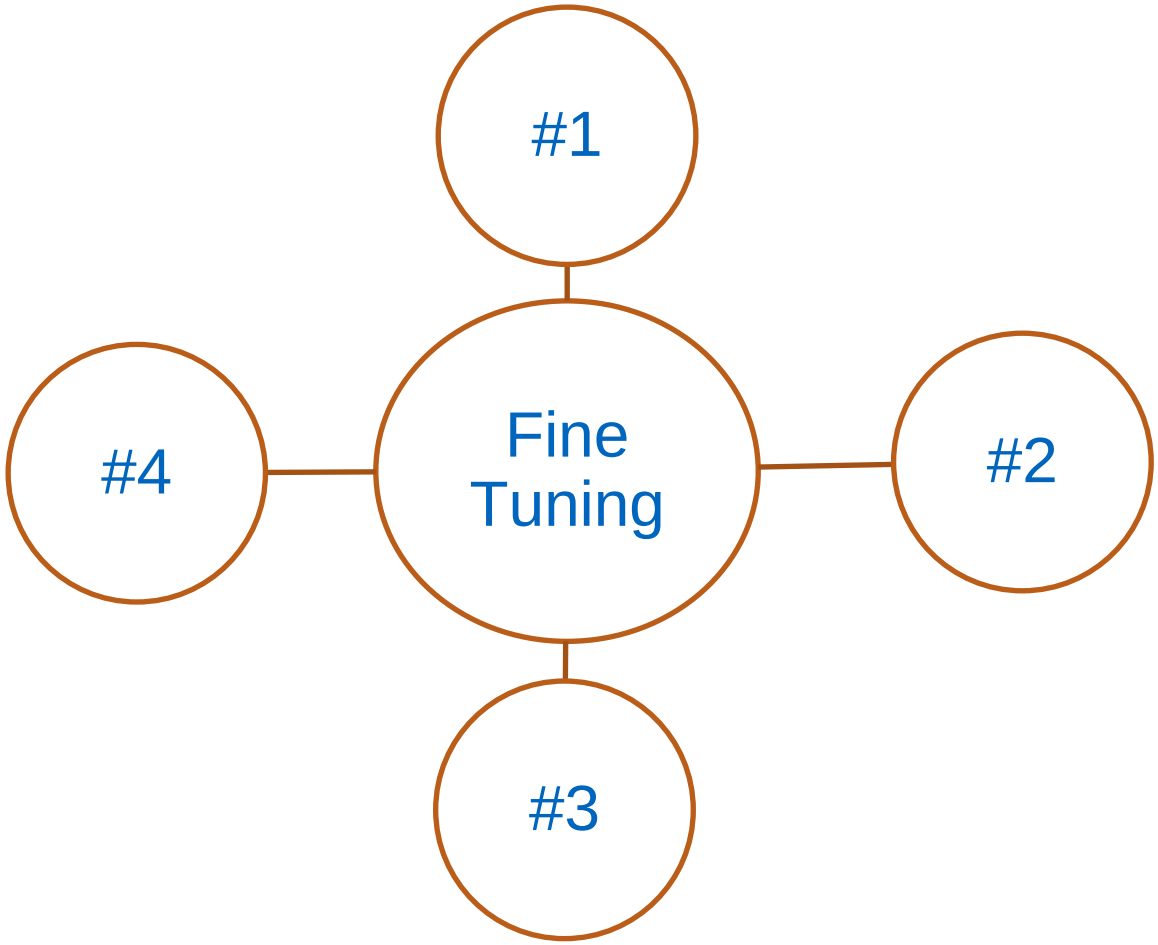


#3:
Right hand is detected as left hand and vice versa

#4
Small part of an object is ignored

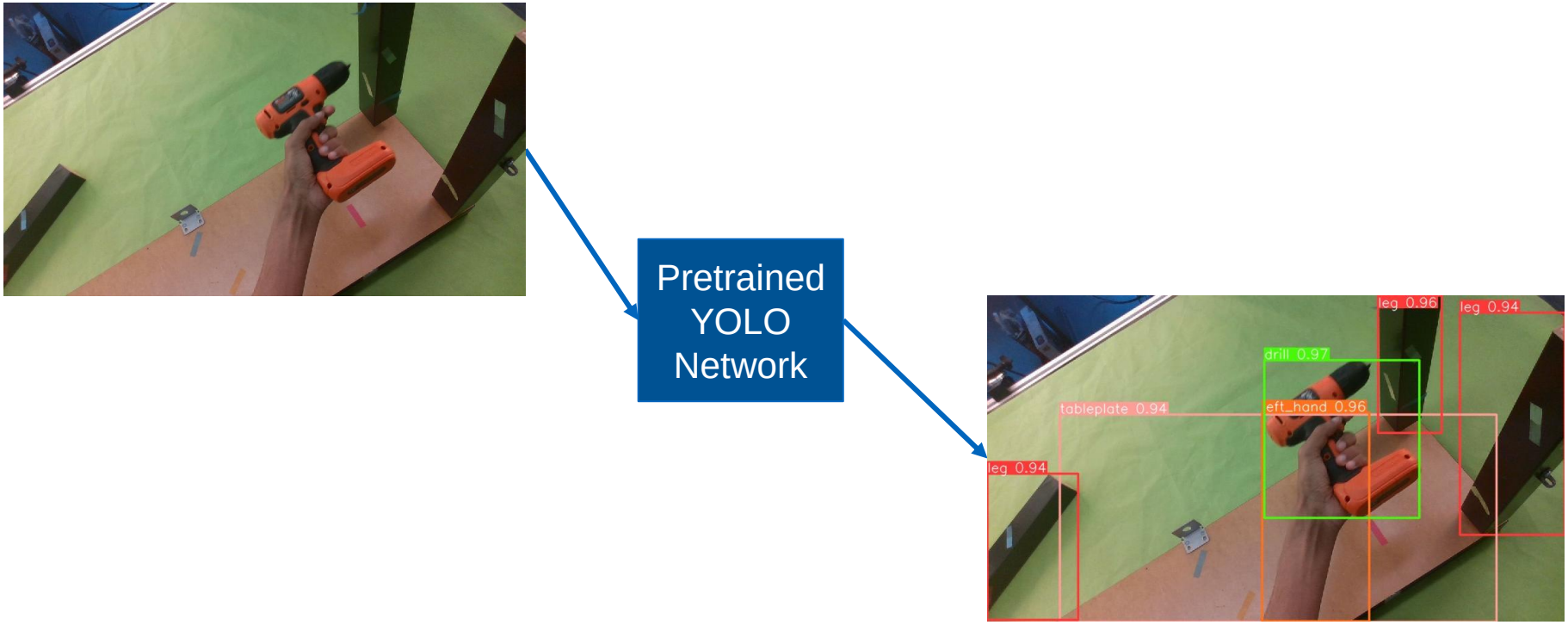


YOLOv8 - Solutions



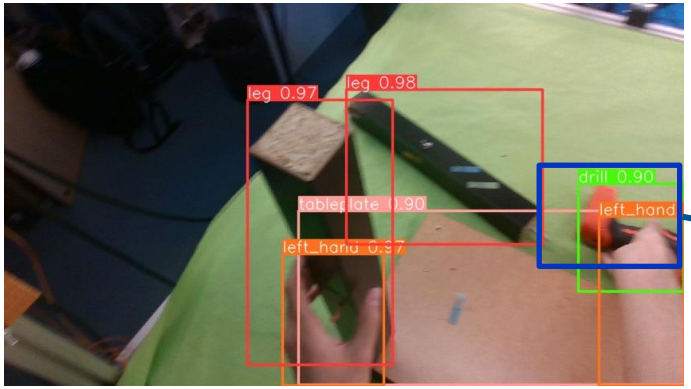
YOLOv8 - Solutions

- Step 1: Use YOLO as an annotator



YOLOv8 - Solutions

- Step 2: Correcting the labels if required



File Options Help s3_t3_00025.txt Save

```
0 0.6480705738067627 0.42756256461143494 0.2886533737182617 0.4040732979774475
0 0.46519413590431213 0.599250078201294 0.2136848270893097 0.691794216632843
2 0.48382794857025146 0.826770544052124 0.14909839630126953 0.3441467583179474
2 0.9375678896903992 0.7781498432159424 0.12486419081687927 0.43969887495040894
5 0.9221975207328796 0.6119017601013184 0.15554805099964142 0.2813313603401184
1 0.7164936661720276 0.7705054879188538 0.5670125484466553 0.4536893367767334
```

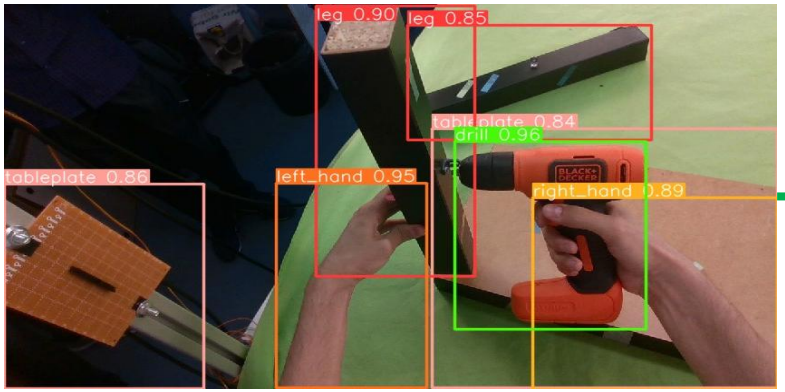
Label Correction

File Options Help s3_t3_00025.txt Save

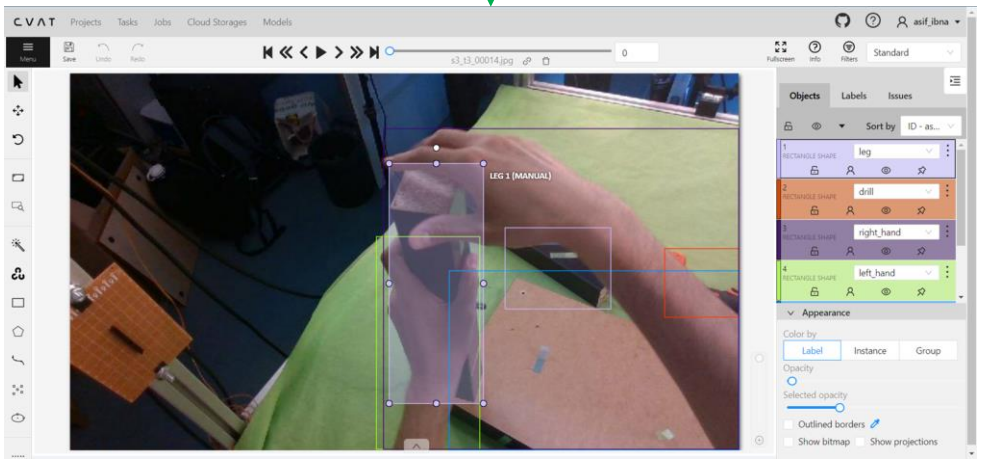
```
0 0.6480705738067627 0.42756256461143494 0.2886533737182617 0.4040732979774475
0 0.46519413590431213 0.599250078201294 0.2136848270893097 0.691794216632843
2 0.48382794857025146 0.826770544052124 0.14909839630126953 0.3441467583179474
3 0.9375678896903992 0.7781498432159424 0.12486419081687927 0.43969887495040894
5 0.9221975207328796 0.6119017601013184 0.15554805099964142 0.2813313603401184
1 0.7164936661720276 0.7705054879188538 0.5670125484466553 0.4536893367767334
```

YOLOv8 - Solutions

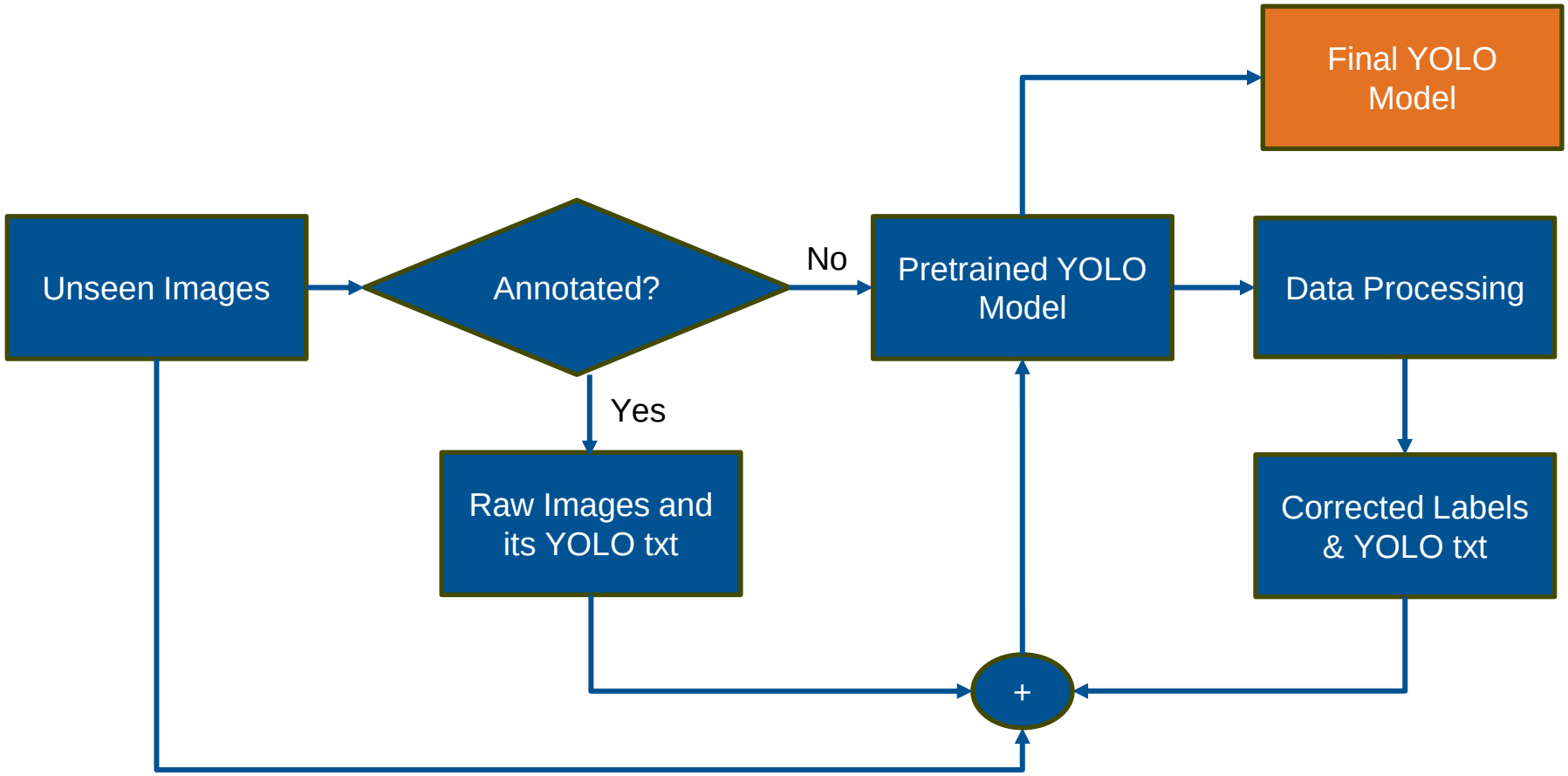
- Step 3: Annotating using CVAT



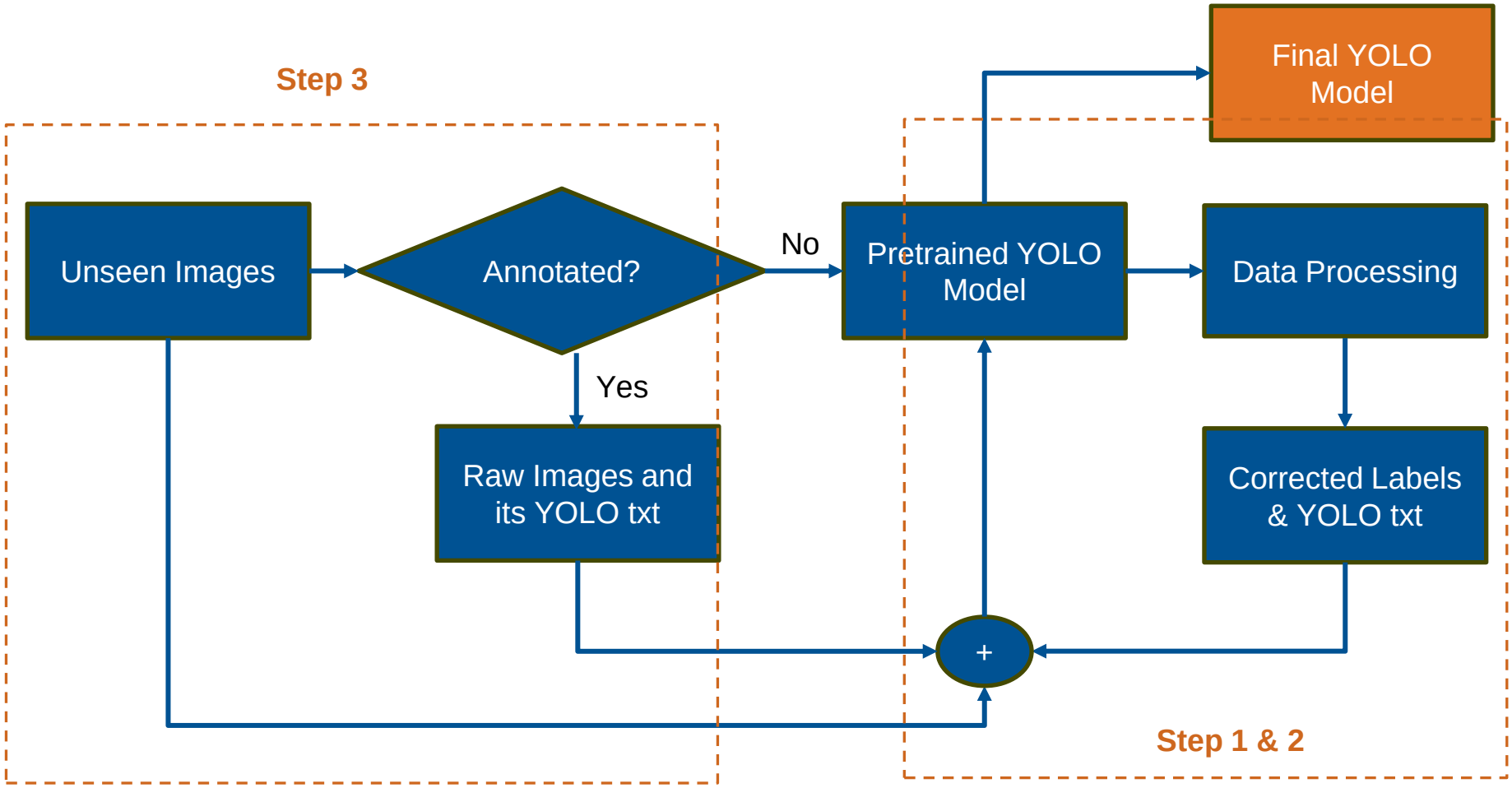
Annotation



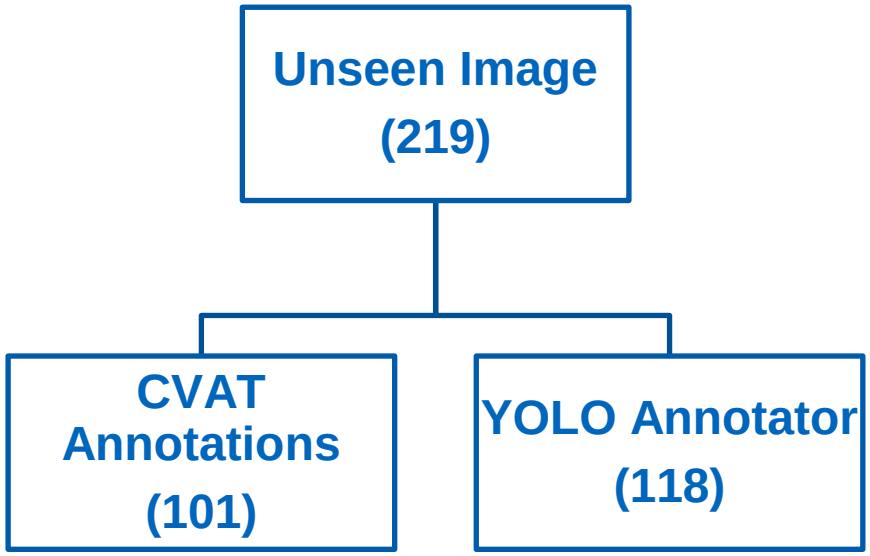
YOLOv8 - Solutions

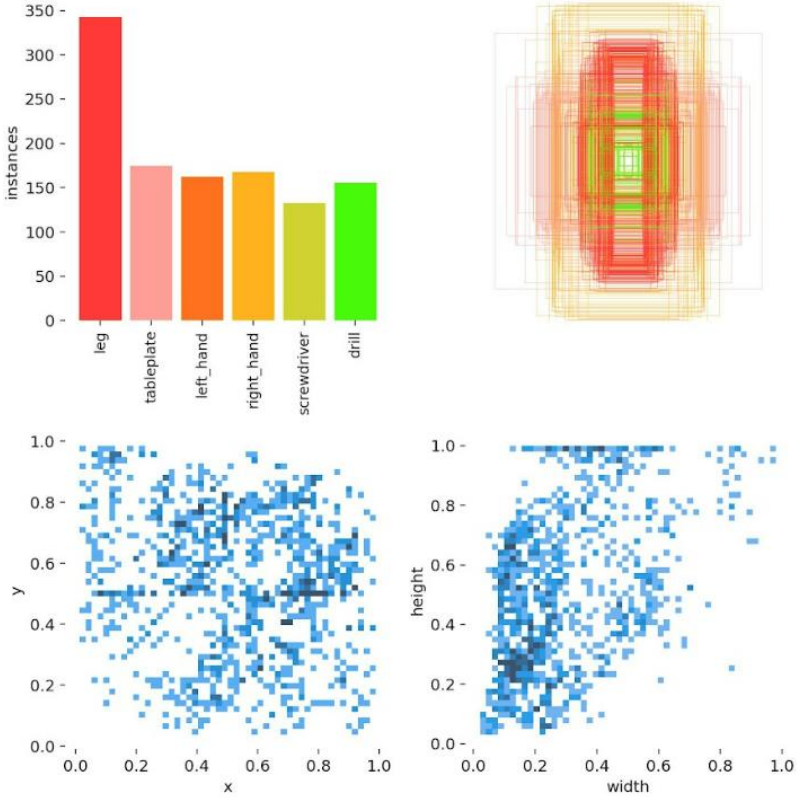


YOLOv8 - Solutions



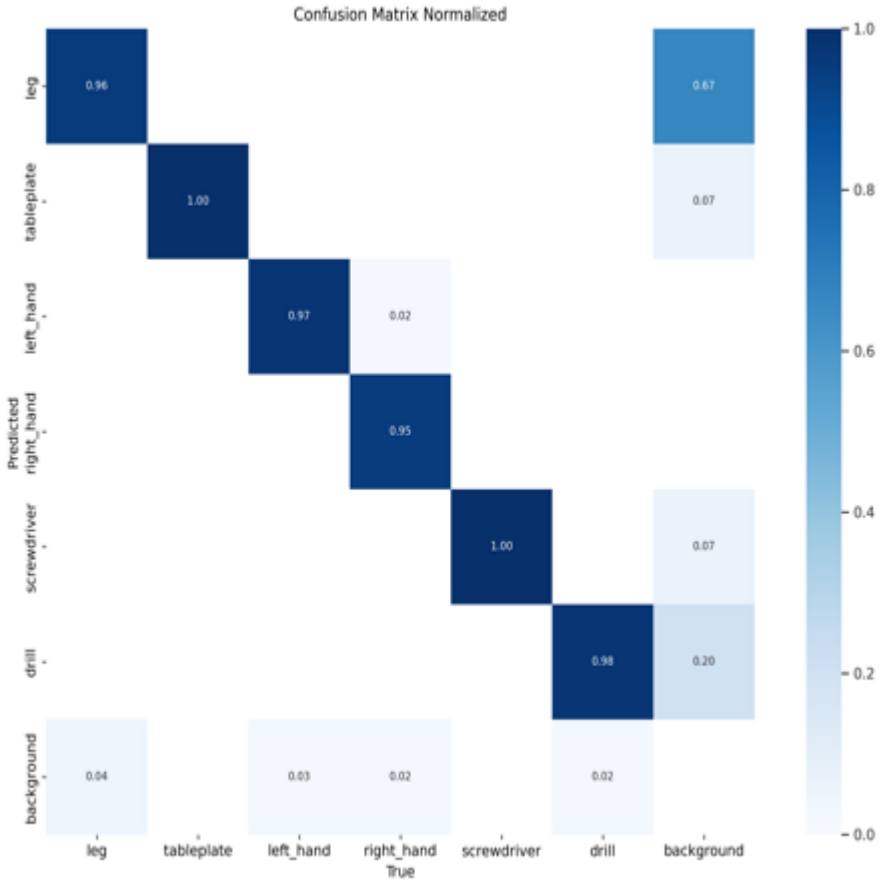
YOLOv8 - FineTuning

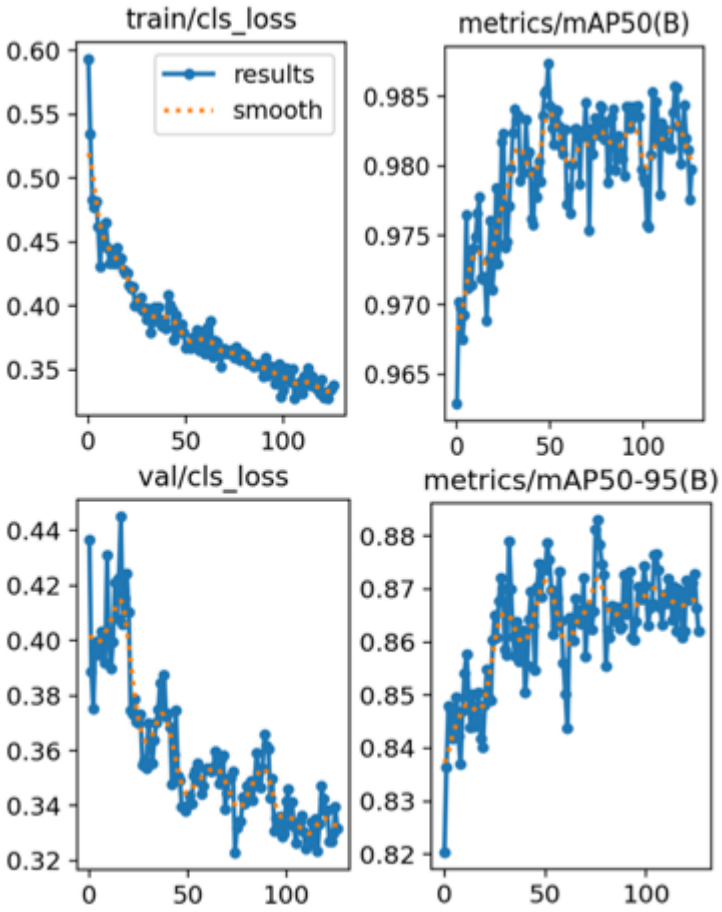




YOLOv8(n)

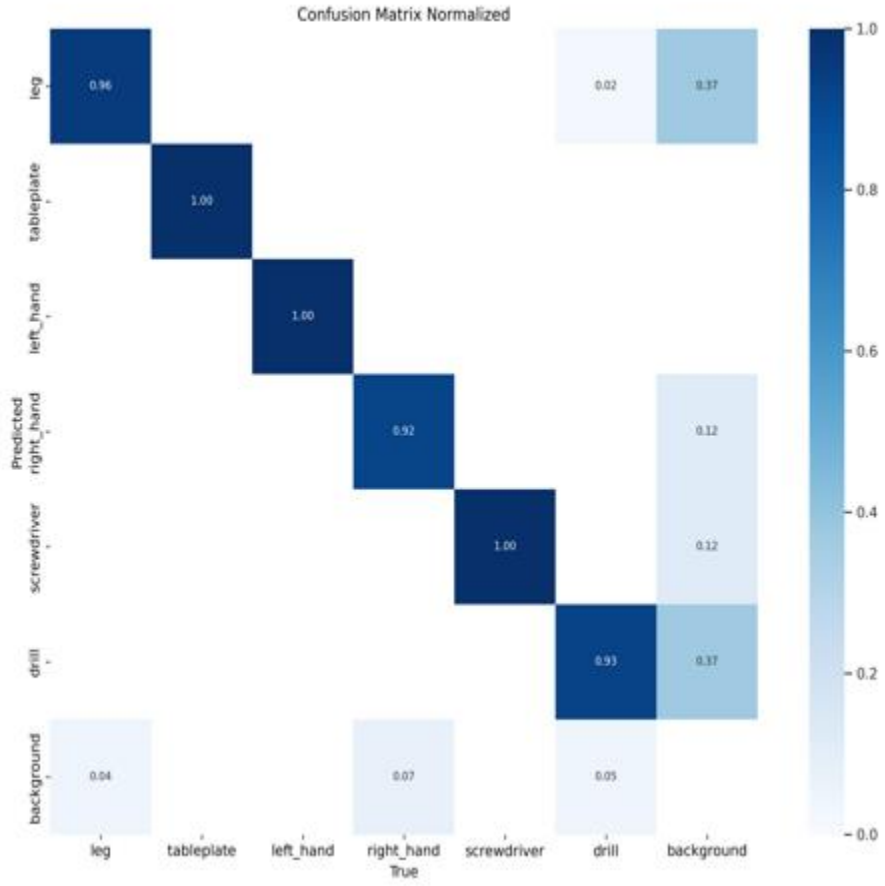
@mAP50: 0.982; @mAP50-95: 0.883

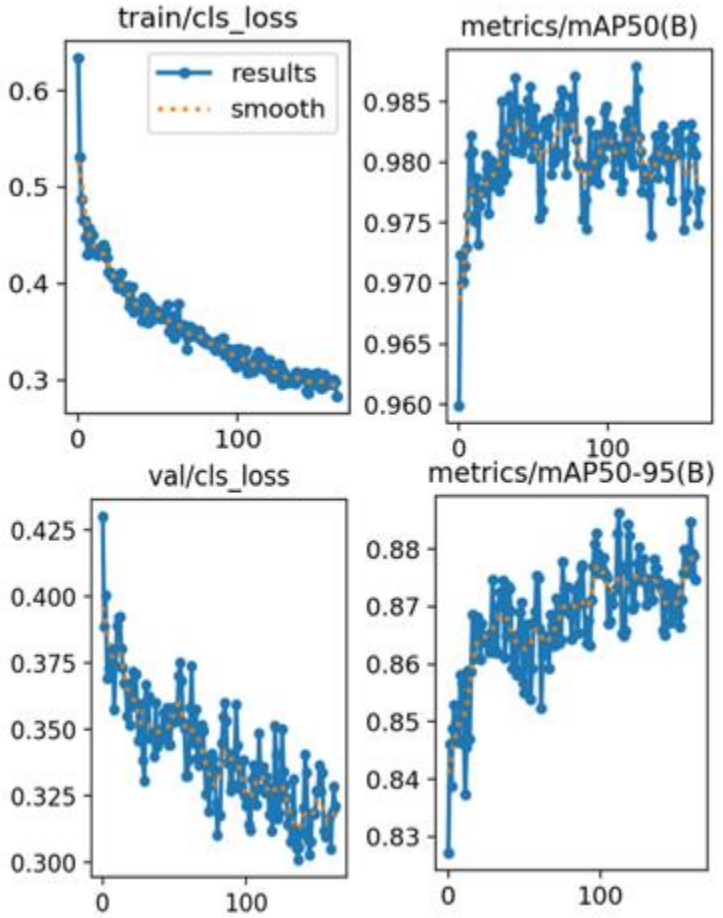




YOLOv8(s)

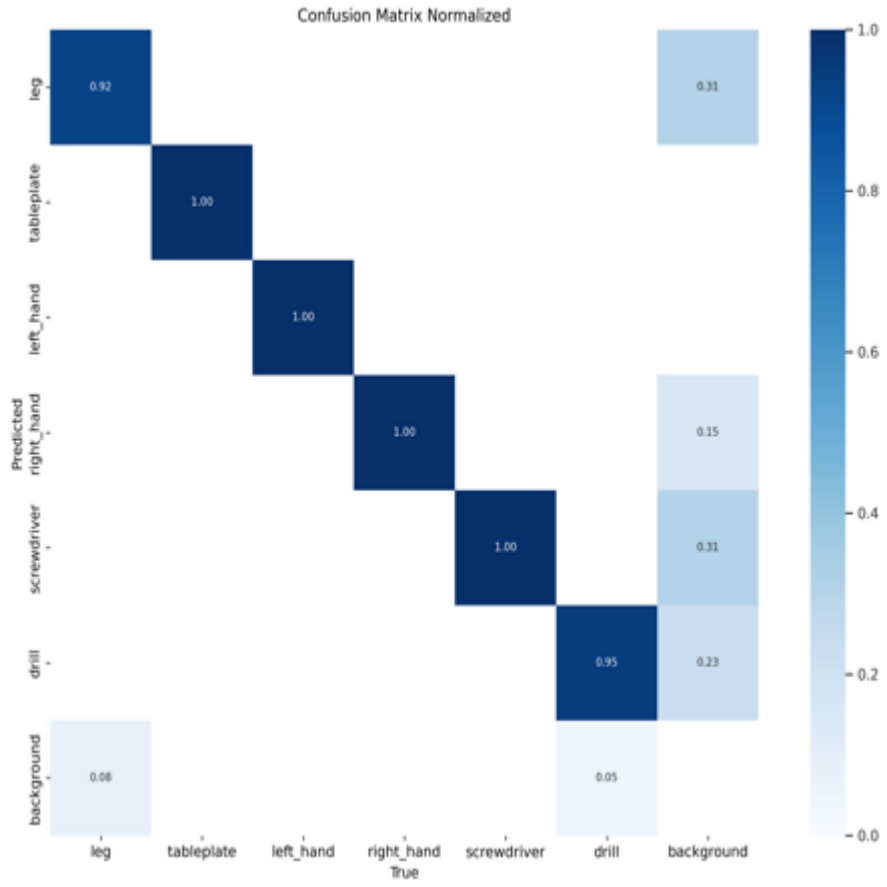
@mAP50: 0.981; @mAP50-95: 0.886

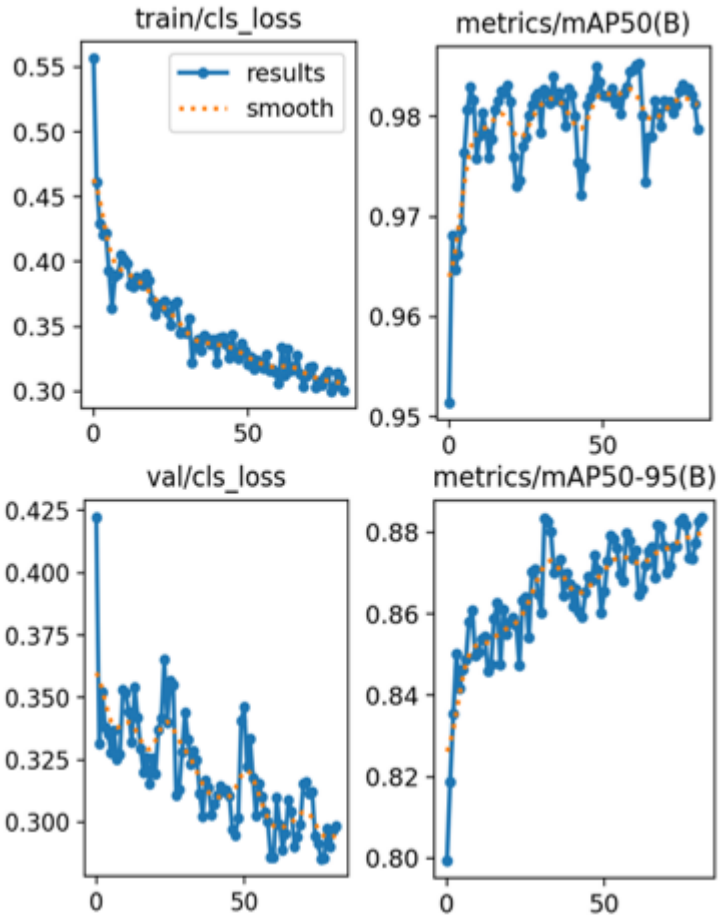




YOLOv8(m)

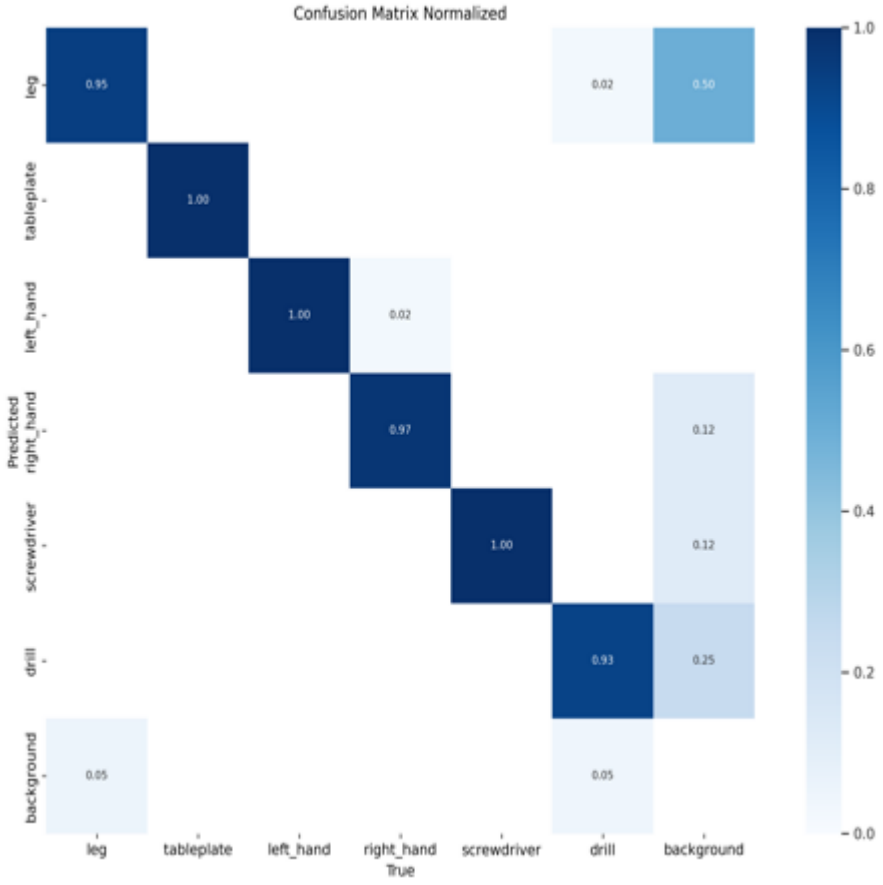
@mAP50: 0.983; @mAP50-95: 0.883

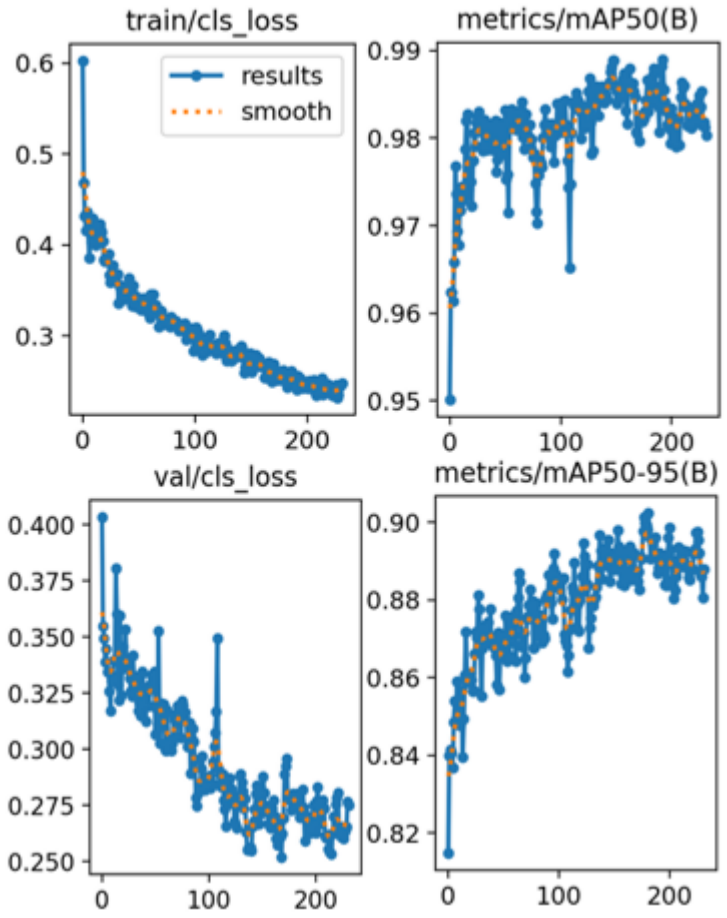




YOLOv8(l)

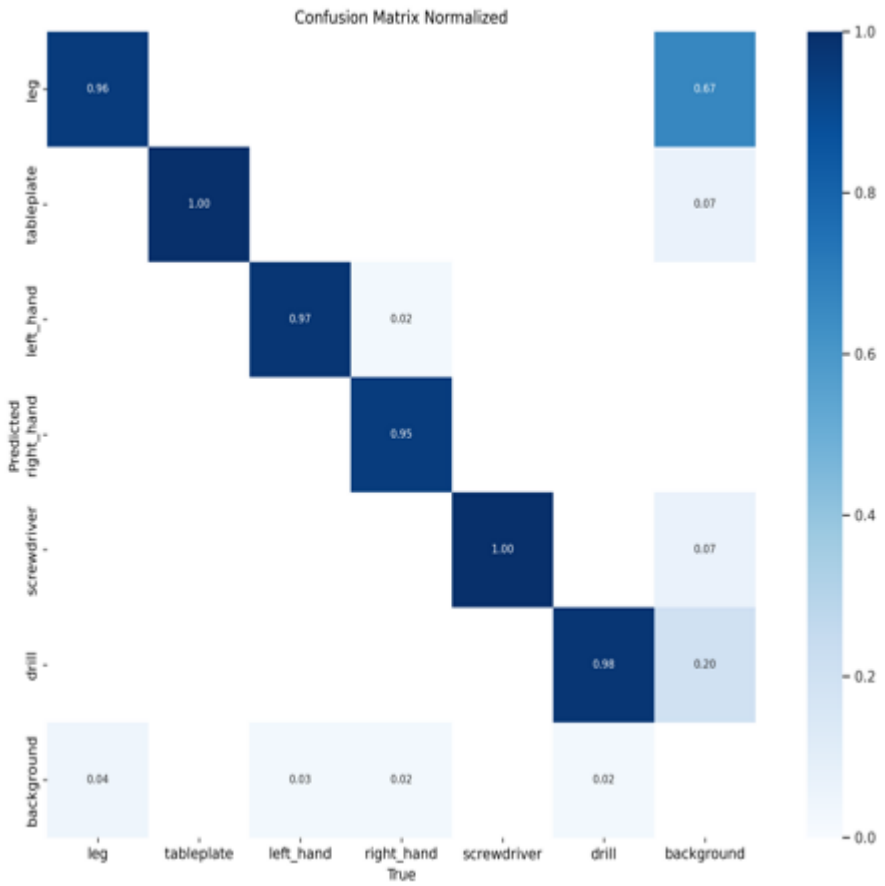
@mAP50: 0.984; @mAP50-95: 0.902

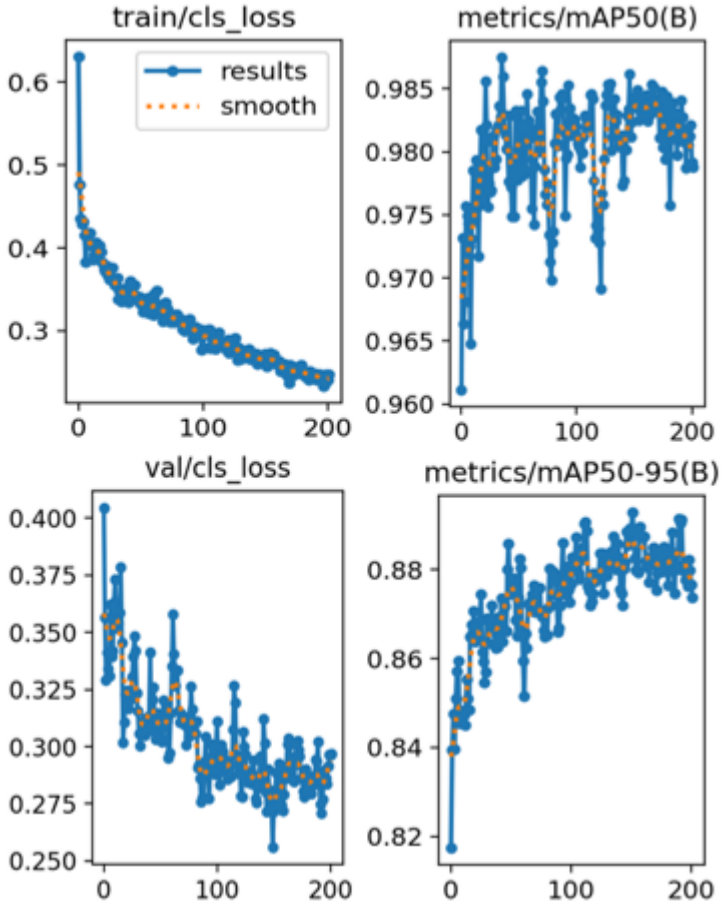




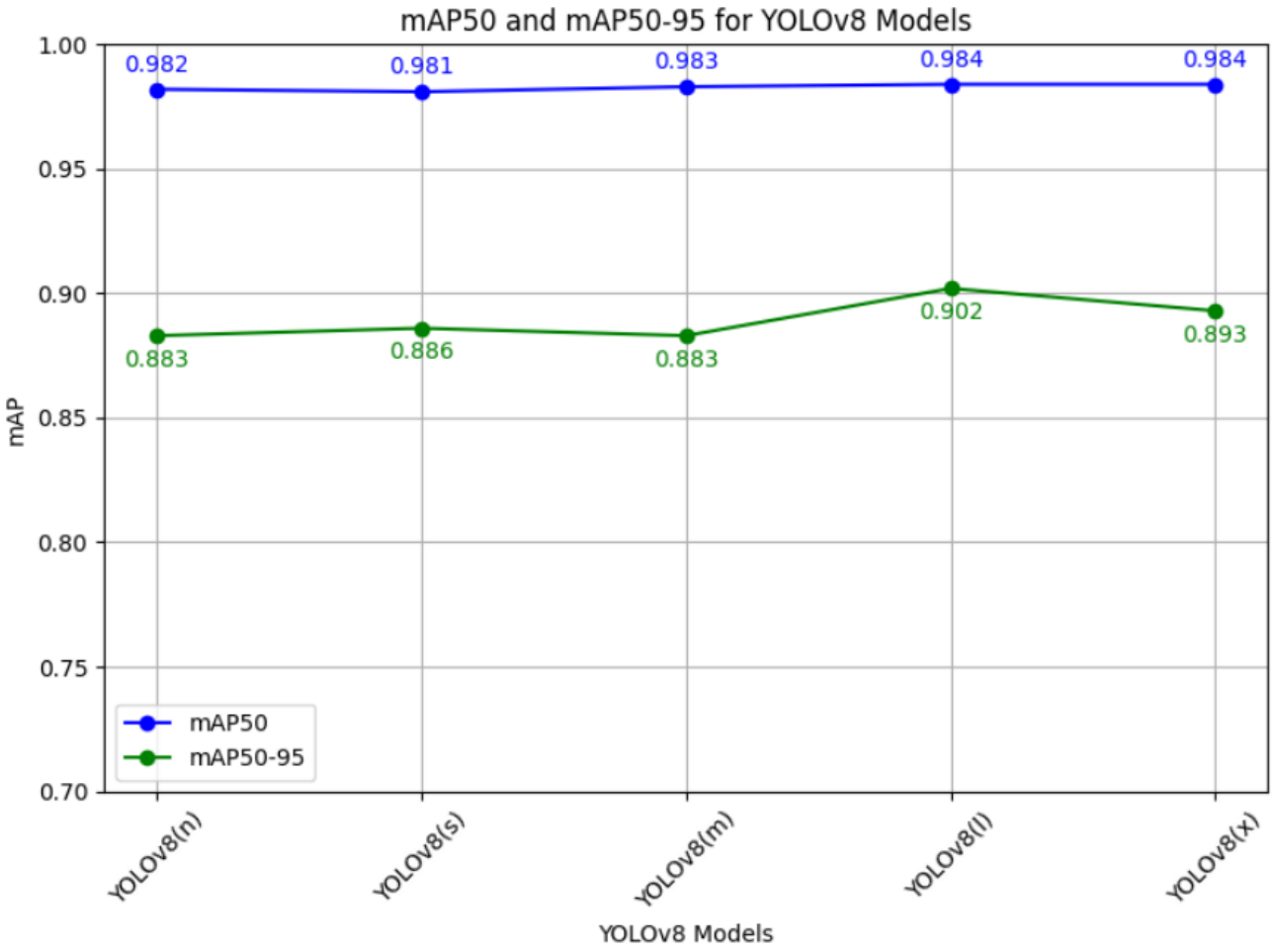
YOLOv8(x)

@mAP50: 0.984; @mAP50-95: 0.893





YOLOv8 – Model Comparison



YOLOv8(n) – Ray Tune

Parameter	Value Range	Description
lr0	tune.uniform(1e-5, 1e-1)	Initial learning rate
lrf	tune.uniform(0.01, 1.0)	Final learning rate factor
momentum	tune.uniform(0.6, 0.98)	Momentum
weight_decay	tune.uniform(0.0, 0.001)	Weight decay
warmup_epochs	tune.uniform(0.0, 5.0)	Warmup epochs
warmup_momentum	tune.uniform(0.0, 0.95)	Warmup momentum
box	tune.uniform(0.02, 0.2)	Box loss weight
cls	tune.uniform(0.2, 4.0)	Class loss weight
hsv_h	tune.uniform(0.0, 0.1)	Hue augmentation range
hsv_s	tune.uniform(0.0, 0.9)	Saturation augmentation range
hsv_v	tune.uniform(0.0, 0.9)	Value (brightness) augmentation range

YOLOv8(n) – Ray Tune

Parameter	Value Range	Description
degrees	tune.uniform(0.0, 45.0)	Rotation augmentation range (degrees)
translate	tune.uniform(0.0, 0.9)	Translation augmentation range
scale	tune.uniform(0.0, 0.9)	Scaling augmentation range
shear	tune.uniform(0.0, 10.0)	Shear augmentation range (degrees)
perspective	tune.uniform(0.0, 0.001)	Perspective augmentation range
flipud	tune.uniform(0.0, 1.0)	Vertical flip augmentation probability
fliplr	tune.uniform(0.0, 1.0)	Horizontal flip augmentation probability
mosaic	tune.uniform(0.0, 1.0)	Mosaic augmentation probability
mixup	tune.uniform(0.0, 1.0)	Mixup augmentation probability
copy_paste	tune.uniform(0.0, 1.0)	Copy-paste augmentation probability

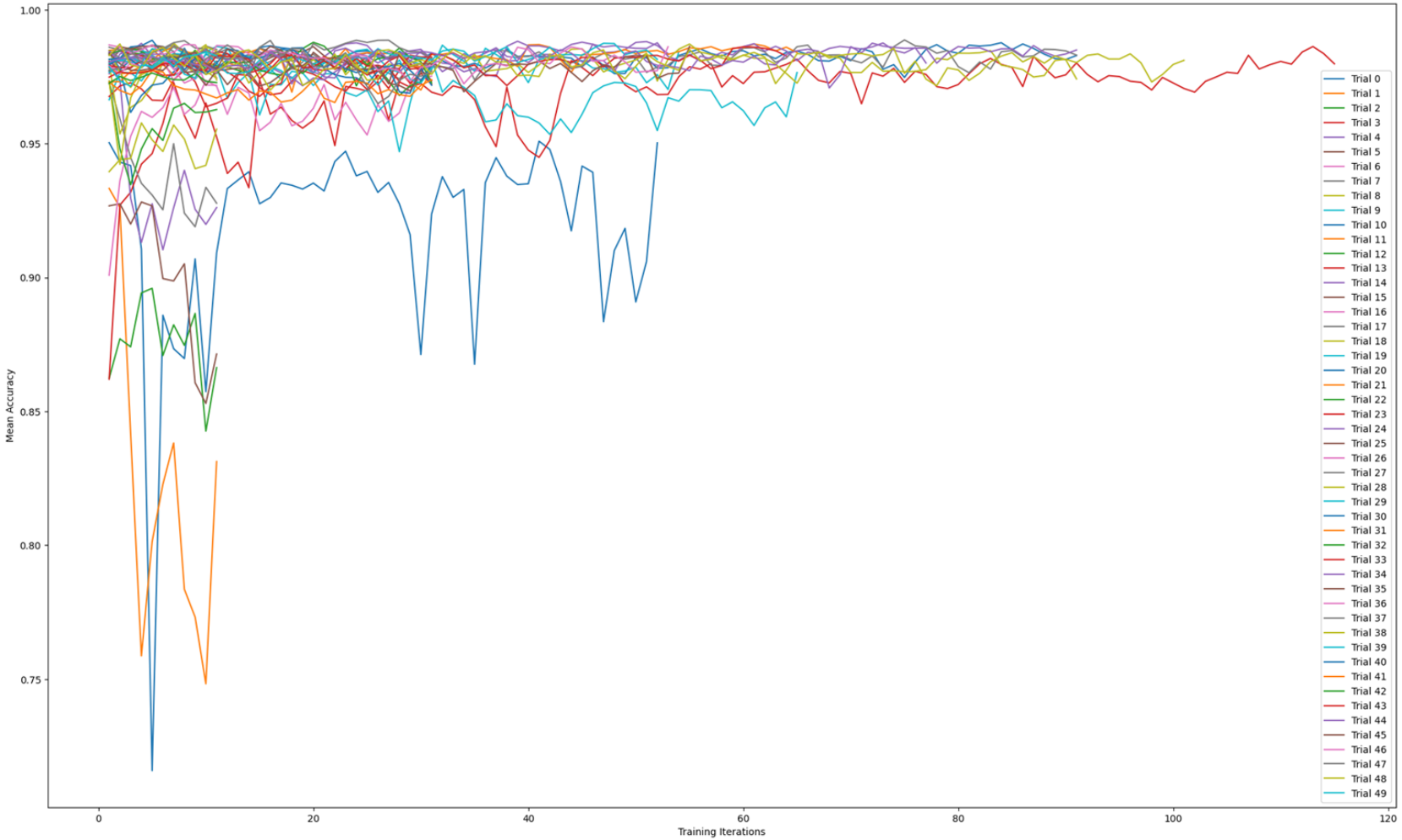
YOLOv8(n) – Ray Tune

Number of trials: 50/50 (50 TERMINATED)

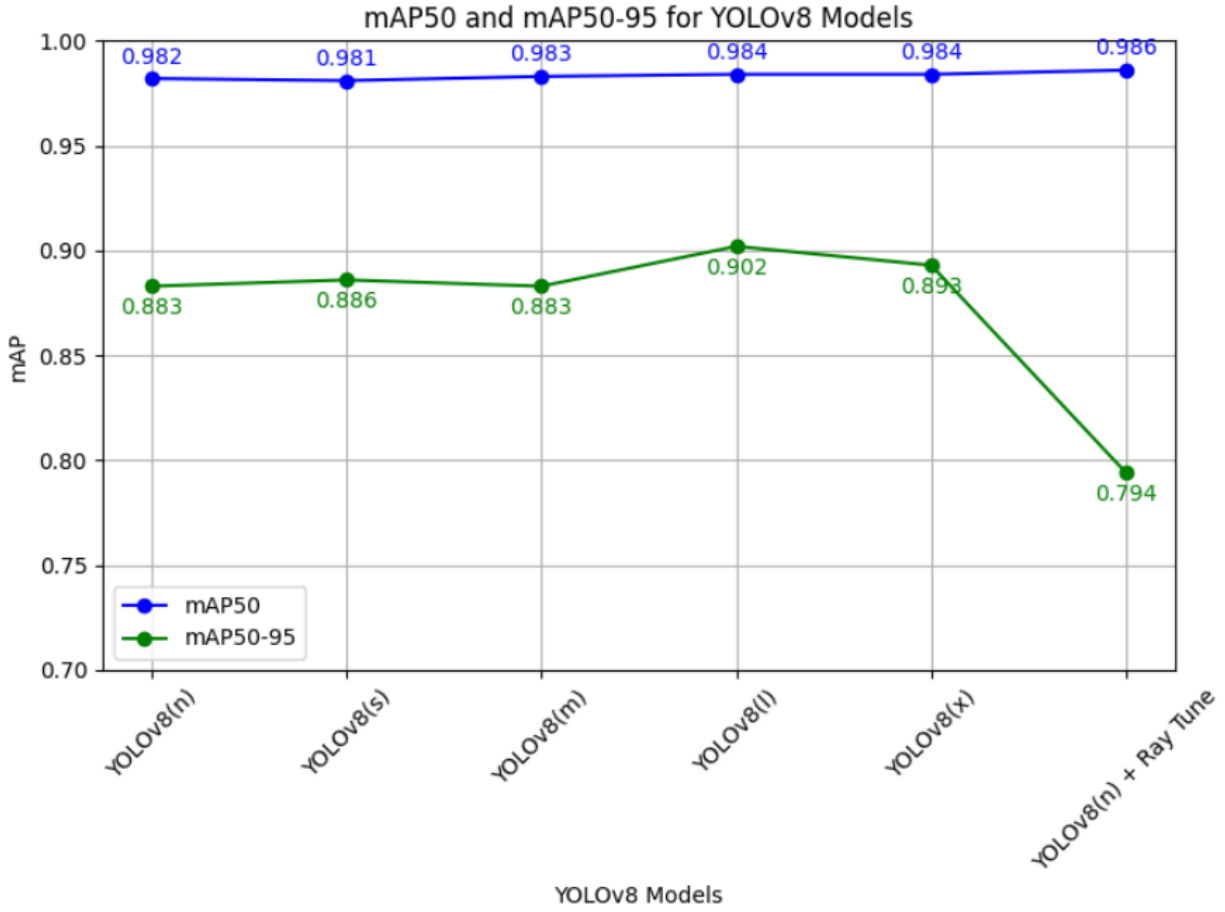
Trial name	status	loc	box	cls	copy_paste	degrees	fliplr	flipud	hsv_h	hsv_s	hsv_v	lr0	lrf	mixup
_tune_876e0_00000	TERMINATED	172.28.0.12:5586	0.0308491	2.67808	0.148072	8.10067	0.884388	0.502116	0.0560387	0.340356	0.269315	0.0749731	0.876187	0.705854
_tune_876e0_00001	TERMINATED	172.28.0.12:5586	0.1123	1.42705	0.10717	39.1486	0.715363	0.648429	0.00773225	0.580834	0.0618743	0.0434468	0.124369	0.0265478
_tune_876e0_00002	TERMINATED	172.28.0.12:5586	0.184055	2.94526	0.815453	29.9873	0.112157	0.589841	0.0481797	0.290138	0.5243	0.0914769	0.579648	0.983074
_tune_876e0_00003	TERMINATED	172.28.0.12:5586	0.0319303	1.957	0.409456	0.504057	0.545156	0.747147	0.0366923	0.676597	0.506473	0.0356369	0.141978	0.620928
_tune_876e0_00004	TERMINATED	172.28.0.12:5586	0.0627681	3.43554	0.398831	28.4371	0.776463	0.337766	0.0629217	0.521903	0.434754	0.0133701	0.898873	0.96381
_tune_876e0_00005	TERMINATED	172.28.0.12:5586	0.157822	2.13706	0.754599	40.5805	0.851354	0.966521	0.0402472	0.867545	0.348341	0.0370528	0.515474	0.989295
_tune_876e0_00006	TERMINATED	172.28.0.12:5586	0.0836739	0.515365	0.236861	12.679	0.280823	0.254015	0.0749082	0.0240684	0.113385	0.0626061	0.595367	0.423919
_tune_876e0_00007	TERMINATED	172.28.0.12:5586	0.0542081	2.01619	0.930754	42.2962	0.733341	0.71928	0.00139835	0.301458	0.665158	0.0378952	0.581109	0.99613
_tune_876e0_00008	TERMINATED	172.28.0.12:5586	0.0644011	3.43757	0.556843	30.0993	0.0431666	0.7785	0.0853695	0.682555	0.656156	0.0777113	0.76317	0.599425
_tune_876e0_00009	TERMINATED	172.28.0.12:5586	0.115263	1.21823	0.00697093	13.2165	0.503885	0.878564	0.0855422	0.236735	0.218289	0.0952504	0.681611	0.632325
_tune_876e0_00010	TERMINATED	172.28.0.12:5586	0.103255	0.232357	0.714846	0.714796	0.266618	0.138945	0.0673746	0.311719	0.786031	0.0266273	0.371103	0.797209
_tune_876e0_00011	TERMINATED	172.28.0.12:5586	0.077321	1.27393	0.394513	6.10242	0.0687592	0.993052	0.019998	0.268709	0.788318	0.0940248	0.463977	0.158846
_tune_876e0_00012	TERMINATED	172.28.0.12:5586	0.0565736	3.97883	0.0513276	35.1066	0.174629	0.158307	0.0230361	0.731293	0.265465	0.0771333	0.152914	0.244273
_tune_876e0_00013	TERMINATED	172.28.0.12:5586	0.106866	0.981477	0.97638	25.6549	0.60806	0.845941	0.0234741	0.686472	0.866685	0.047634	0.751935	0.263682
_tune_876e0_00014	TERMINATED	172.28.0.12:5586	0.0712853	2.32054	0.246074	8.89802	0.0681816	0.530592	0.0216749	0.482236	0.223606	0.04629	0.539507	0.319019
_tune_876e0_00015	TERMINATED	172.28.0.12:5586	0.0892804	3.95989	0.511028	10.5013	0.603851	0.113729	0.0354056	0.112345	0.444894	0.0556944	0.796588	0.665444
_tune_876e0_00016	TERMINATED	172.28.0.12:5586	0.130891	0.804049	0.887086	21.0601	0.0699281	0.402088	0.0374927	0.283748	0.556235	0.0167154	0.95716	0.048804
_tune_876e0_00017	TERMINATED	172.28.0.12:5586	0.157496	3.09177	0.0925523	8.70312	0.241531	0.4779	0.0150544	0.593724	0.760892	0.0889354	0.555359	0.72101
_tune_876e0_00018	TERMINATED	172.28.0.12:5586	0.195998	1.1546	0.749977	38.4928	0.169896	0.416021	0.0202739	0.625549	0.522981	0.0747001	0.78606	0.53061
_tune_876e0_00019	TERMINATED	172.28.0.12:5586	0.170206	1.96449	0.0712053	8.45247	0.787285	0.687494	0.0389238	0.897712	0.813846	0.0529134	0.275031	0.456284
_tune_876e0_00020	TERMINATED	172.28.0.12:5586	0.114612	3.78154	0.680071	44.0918	0.274161	0.400462	0.0420003	0.609537	0.771737	0.0140843	0.0214491	0.988412
_tune_876e0_00021	TERMINATED	172.28.0.12:5586	0.156619	2.85651	0.369584	25.9487	0.159386	0.832633	0.0178399	0.277947	0.405069	0.051986	0.425703	0.676492
_tune_876e0_00022	TERMINATED	172.28.0.12:5586	0.189264	1.97674	0.36928	23.3894	0.555912	0.648607	0.0826612	0.119686	0.814911	0.0915184	0.603112	0.413534
_tune_876e0_00023	TERMINATED	172.28.0.12:5586	0.0353554	1.44216	0.00746378	14.28	0.579179	0.786721	0.0500952	0.0106165	0.242253	0.0909723	0.358913	0.760681
_tune_876e0_00024	TERMINATED	172.28.0.12:5586	0.144973	3.7538	0.295548	12.4357	0.16132	0.392212	0.0195925	0.547249	0.280695	0.0334178	0.626882	0.63201

- mAP50 increased from 0.982 to 0.986
- mAP50-95 decreased from 0.883 to 0.794

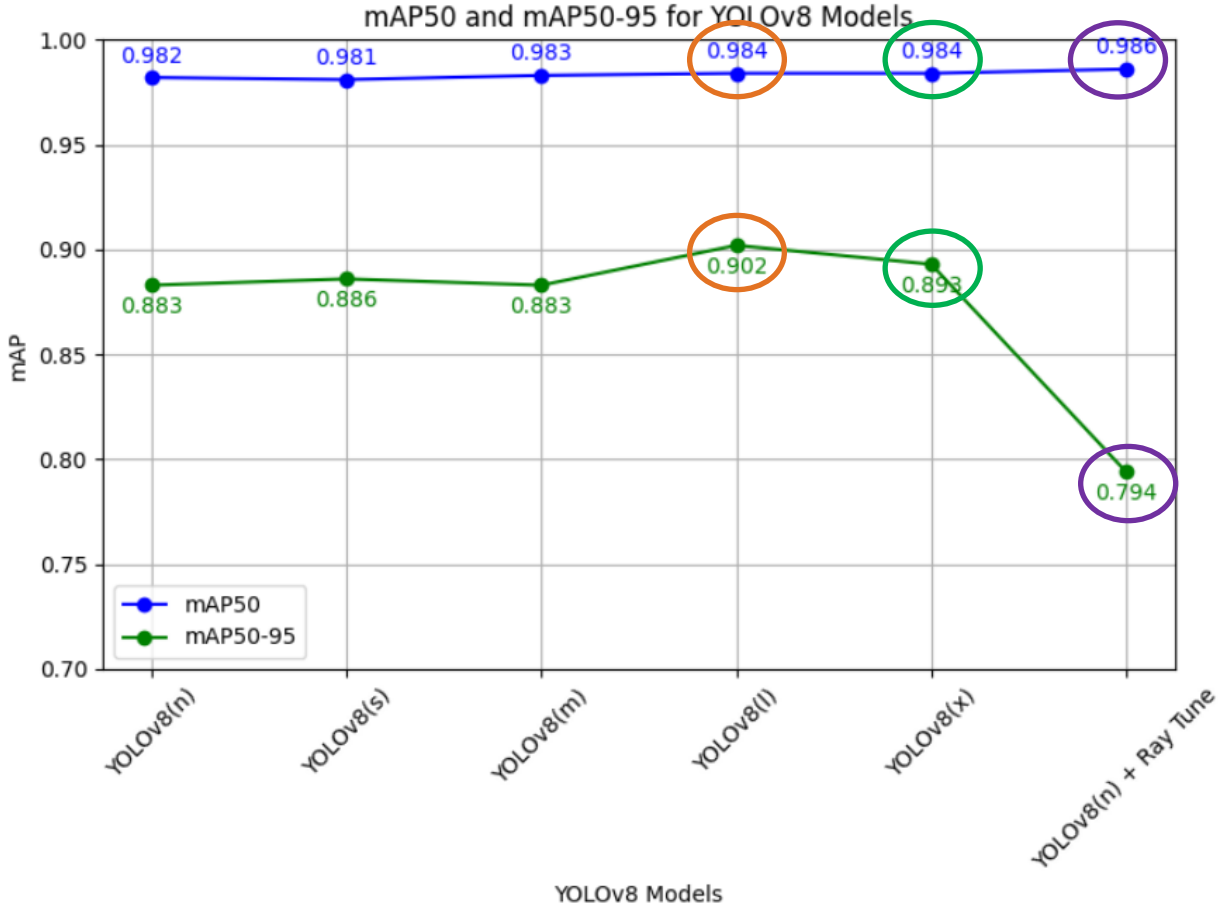
YOLOv8(n) – Ray Tune



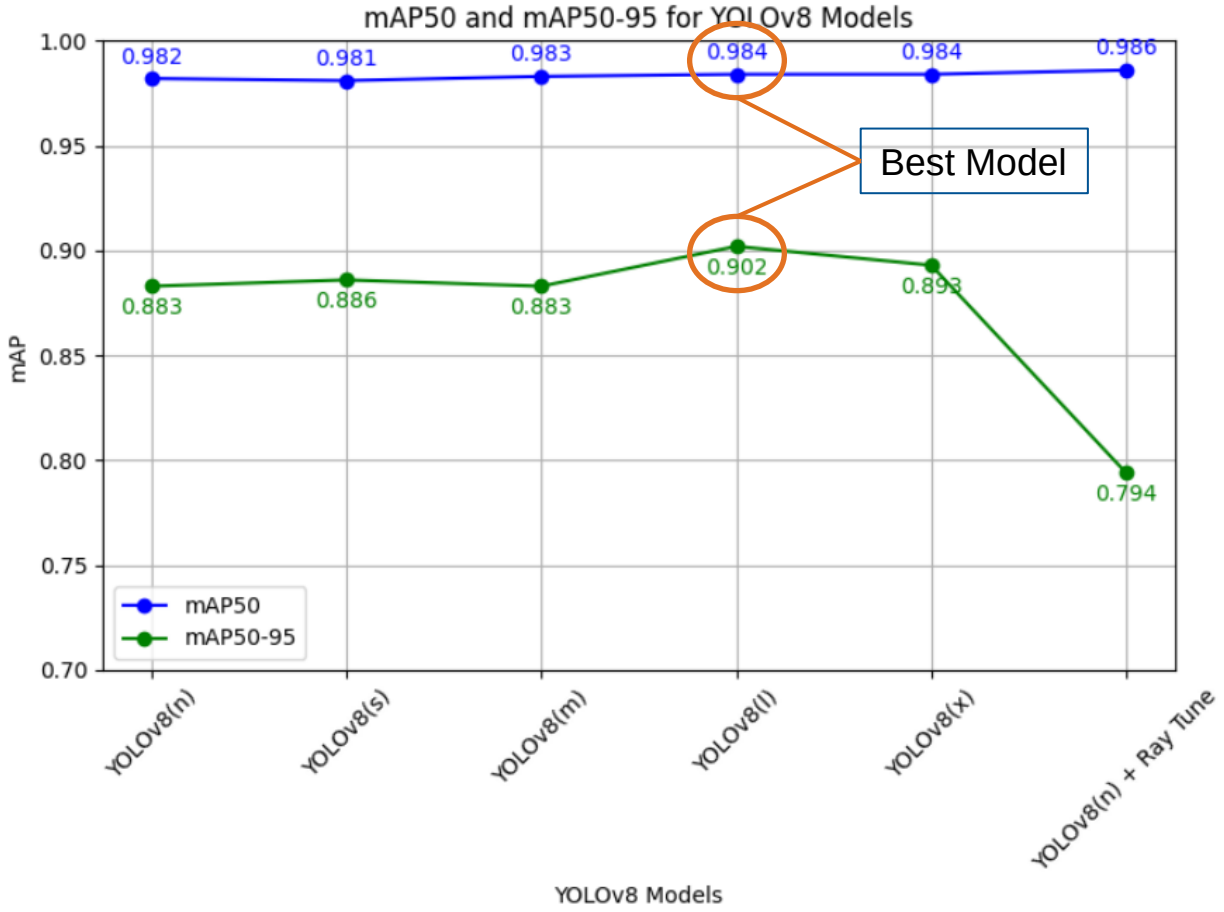
YOLOv8 – Model Comparison



YOLOv8 – Model Comparison



YOLOv8 – Model Comparison



Hand landmarks detection

Zain Amir Zaman

Hand landmarks detection

What is it? Hand landmarks refer to the location of the joints in the hand. Allows for hand pose estimation.

Motivation

- Activities depend on actions of hands (picking tool, rotating leg, etc.).
- Can be an important feature for the activity recognition model.

Tool: MediaPipe library from Google.

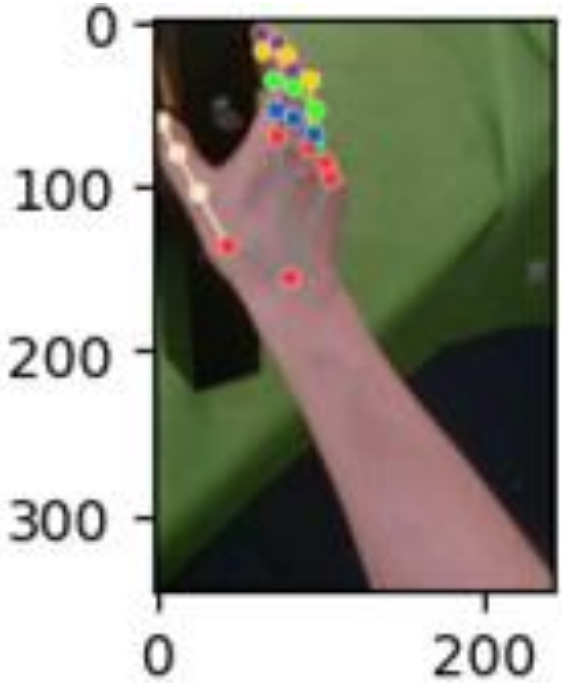


Fig. 1: Landmarks generated by MediaPipe algorithm.

Hand landmarks detection

Problems with existing algorithm

- Failed detection when whole image is processed (Fig. 2).
- Misclassification of hand (Fig. 3).

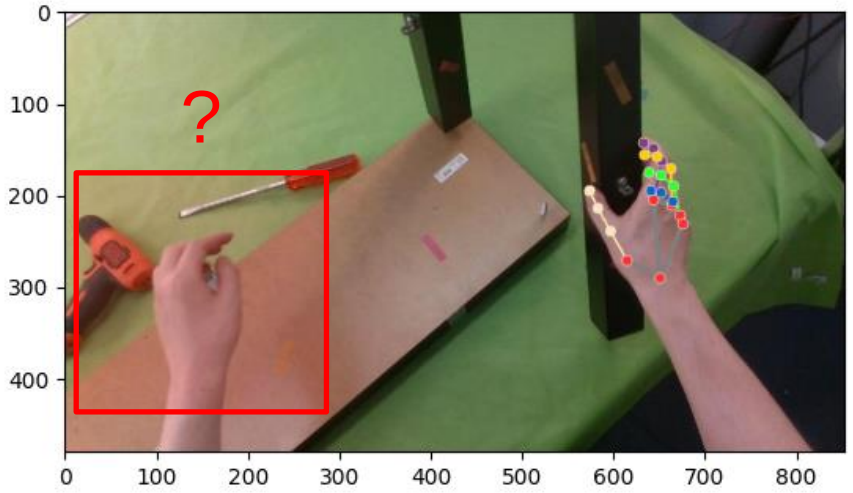
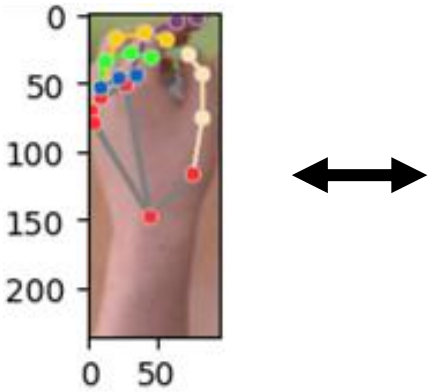


Fig. 2: Missing landmarks on left hand.




```
Handedness: [classification {
  index: 1
  score: 0.9643827676773071
  label: "Right"
}
]
hand_landmarks: landmark {
  ...
```

Fig. 3: Incorrectly labelled hand.

Hand landmarks detection

Solution for both problems: Use the YOLO output!

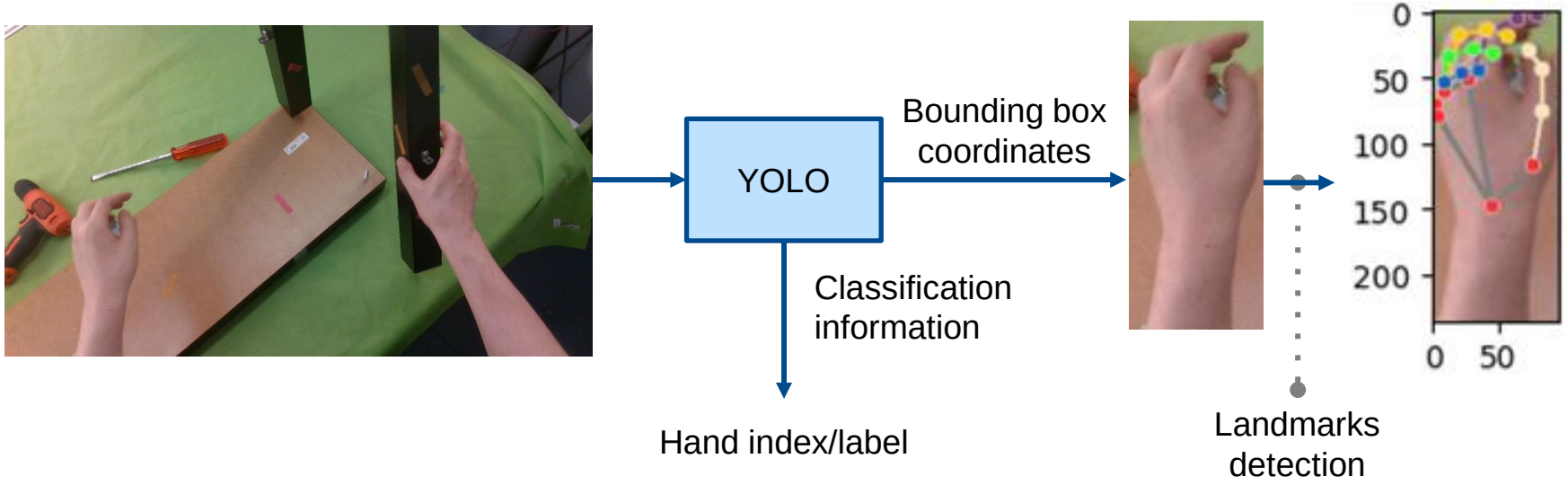


Fig. 4: Using the output of the YOLO model for improving likelihood of detection and correctly classifying right/left hand.

Hand landmarks detection

One further problem…

Cropped images do not guarantee detection.

Solution?

Increase the size of the bounding box until detection occurs.

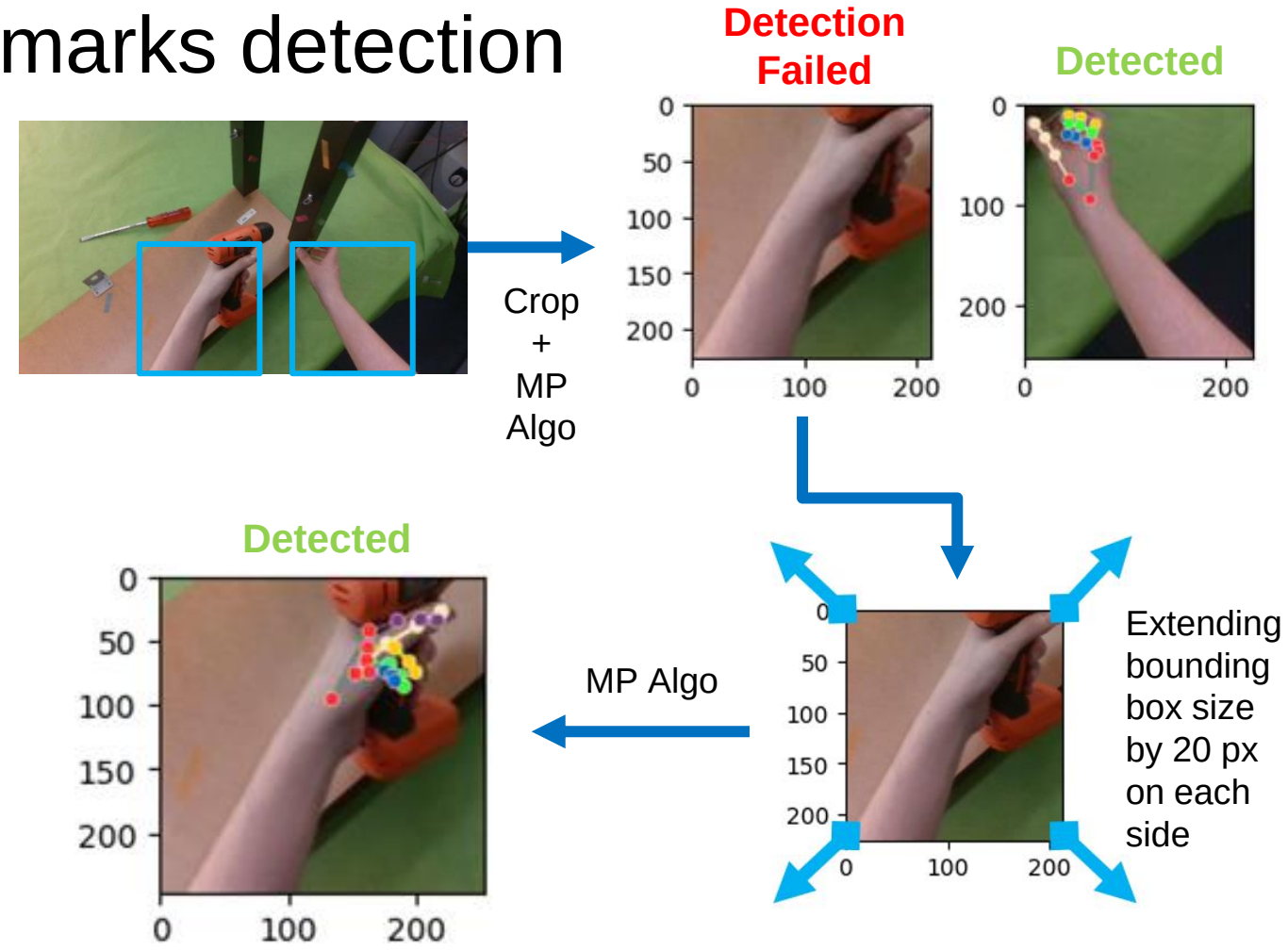


Fig. 5: Expanding the 'field-of-view' of MediaPipe (MP) algorithm.

Hand landmarks detection

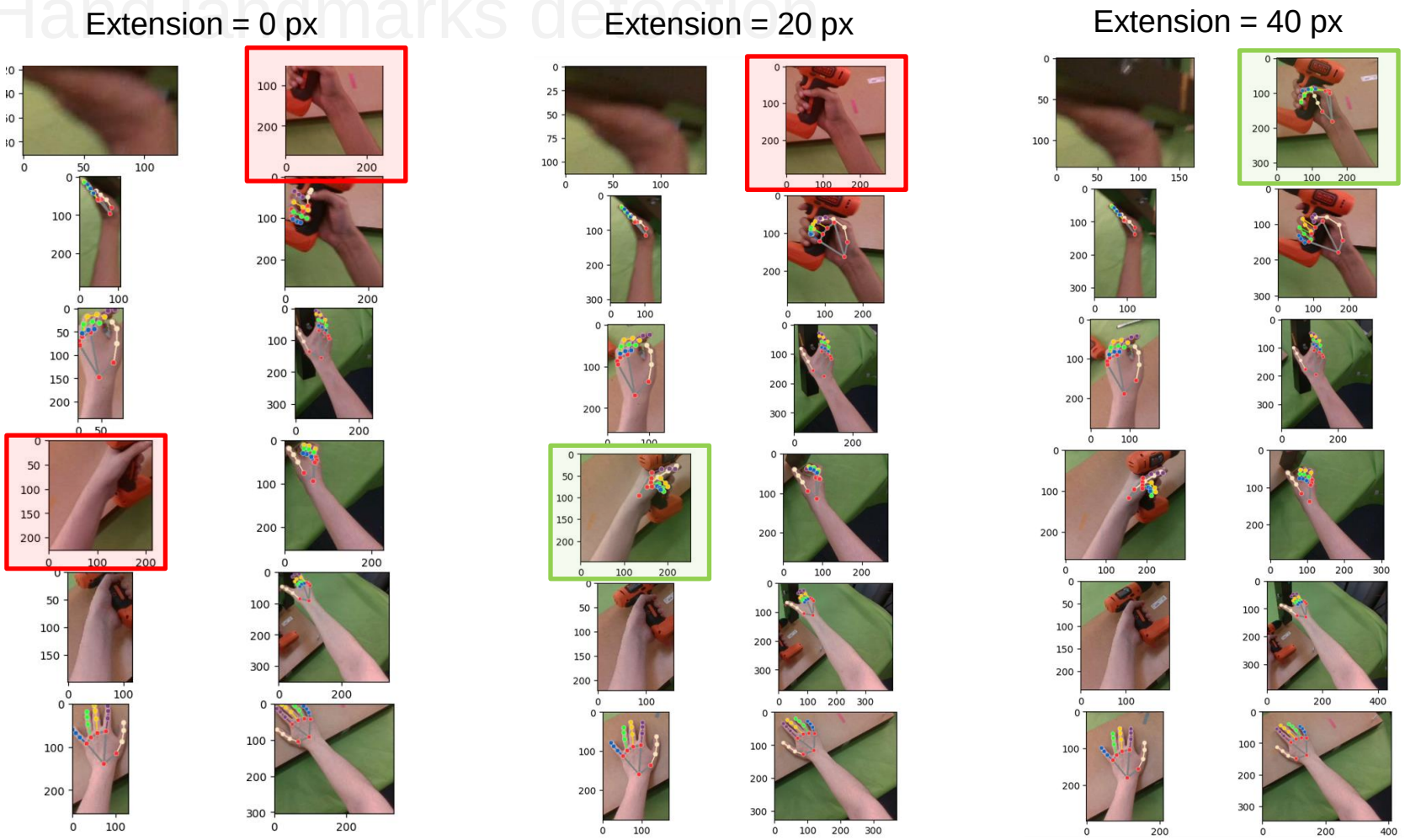


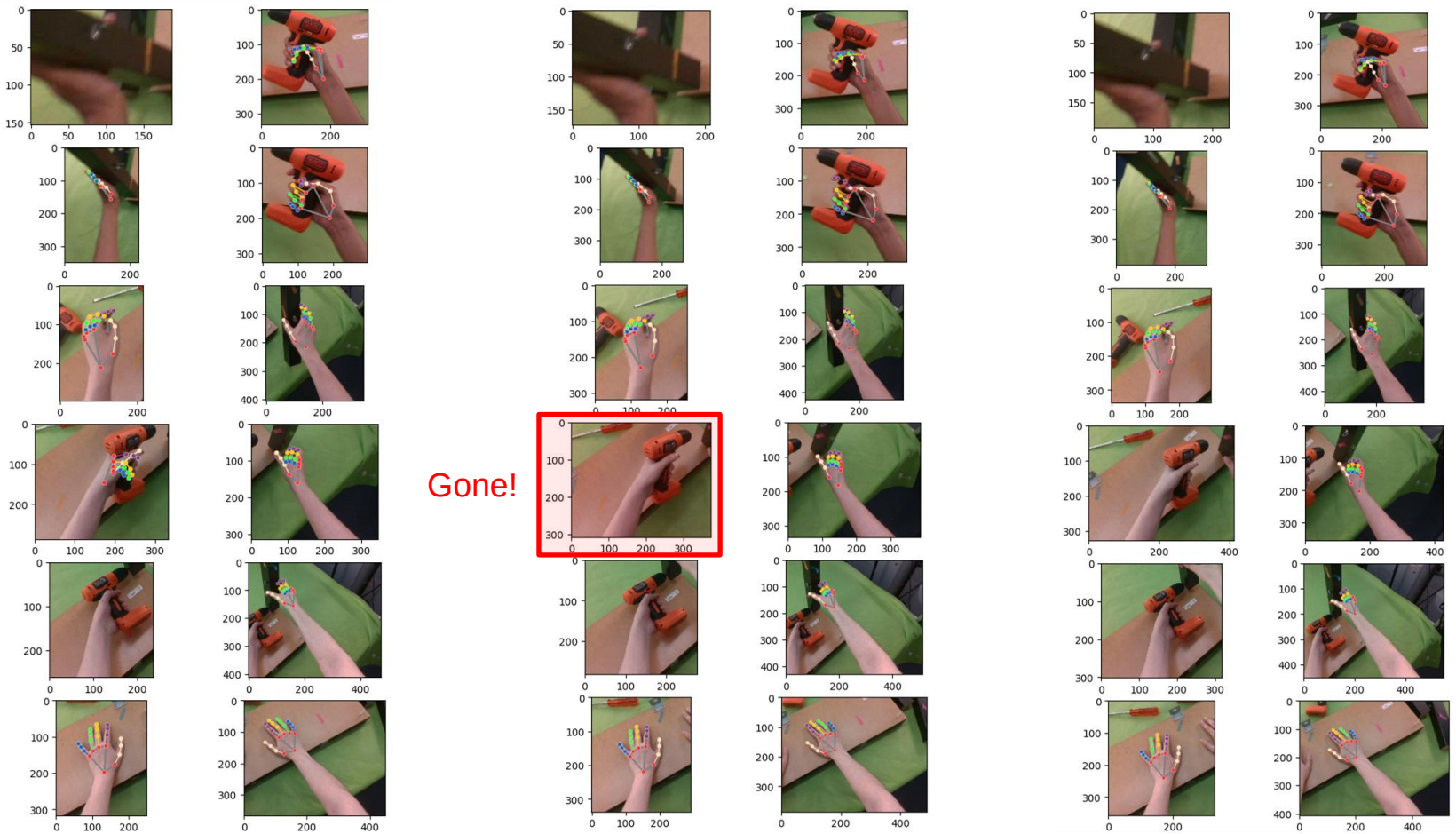
Fig. 6: Impact of different extensions of the bounding box on the detection output.

Hand landmarks detection

Extension = 60 px

Extension = 80 px

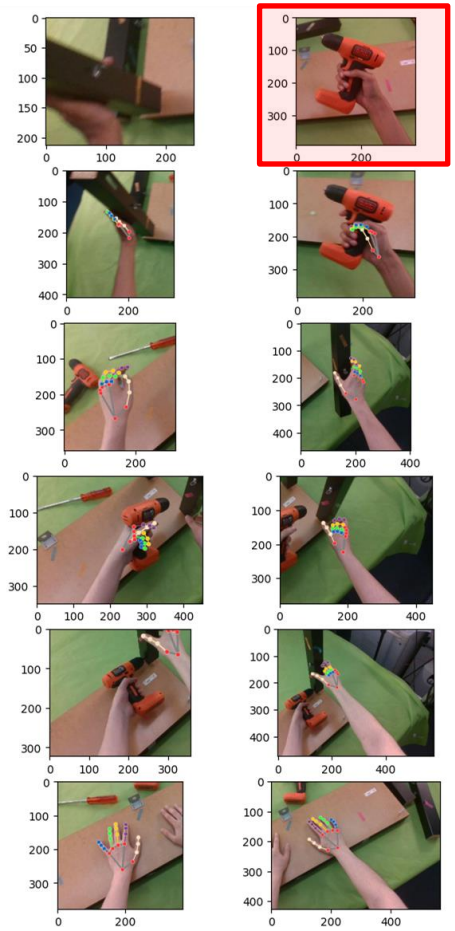
Extension = 100 px



… *continued (Fig. 6)*

Hand landmarks detection

Extension = 120 px



… continued (Fig. 6)

Hand landmarks detection

In summary…

- An initial extension of 20 px can be safely applied to increase chances of first-time detection.
- Successive extensions of 20 px each (up to a max. of 80 px) are sufficient to increase chances of detection.

Hand landmarks detection

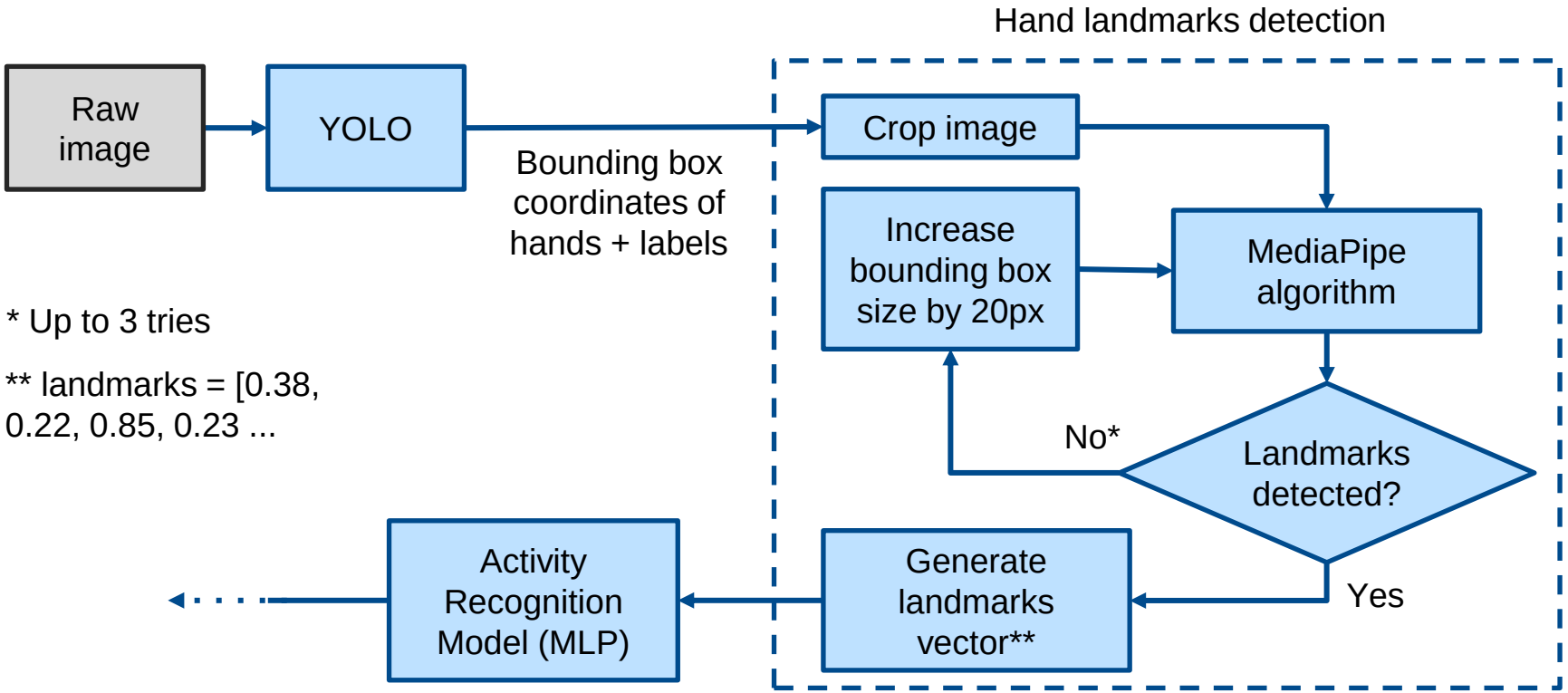
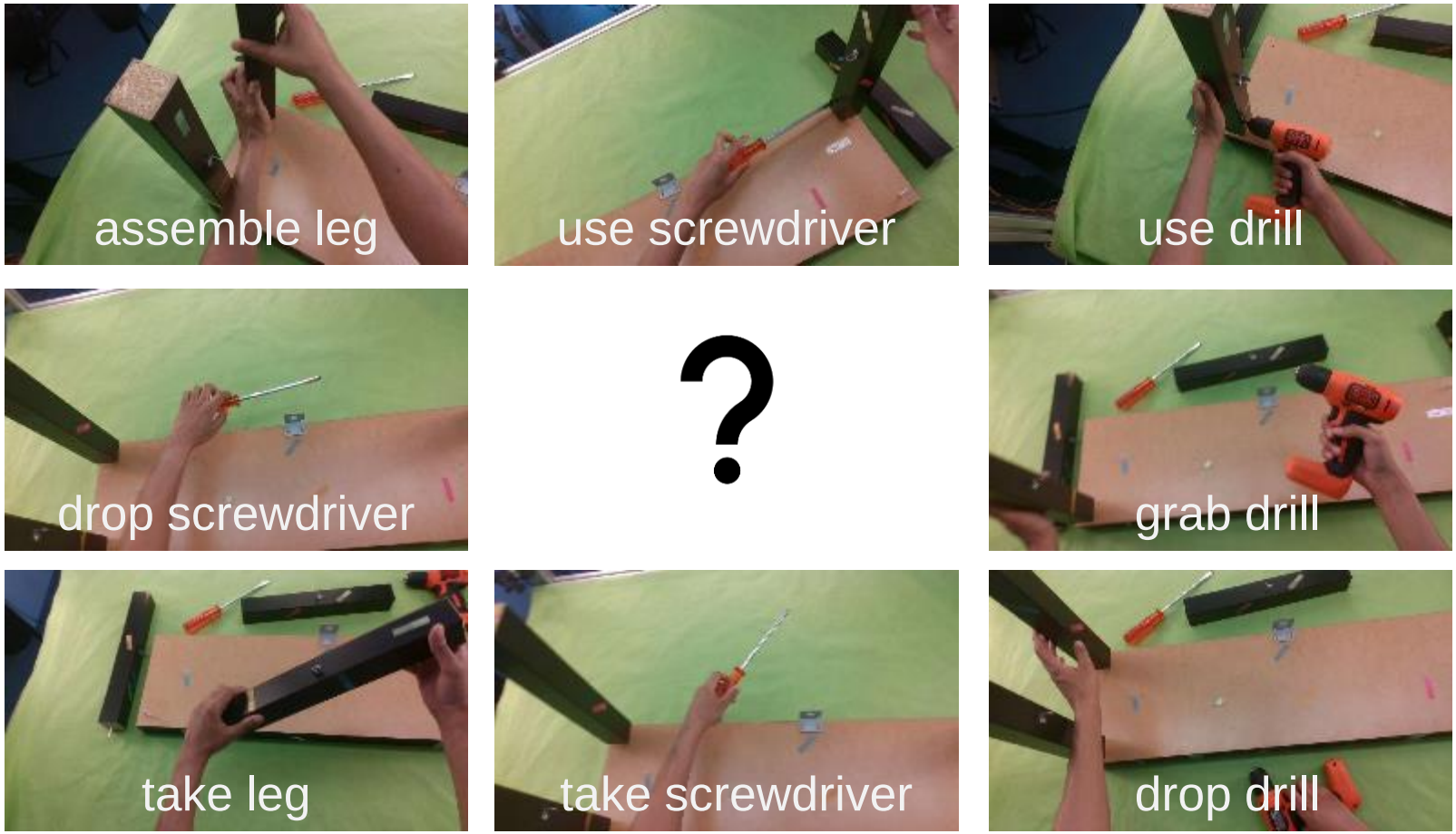


Fig. 7: Final algorithm and integration into project pipeline

Activity Recognition - Overview



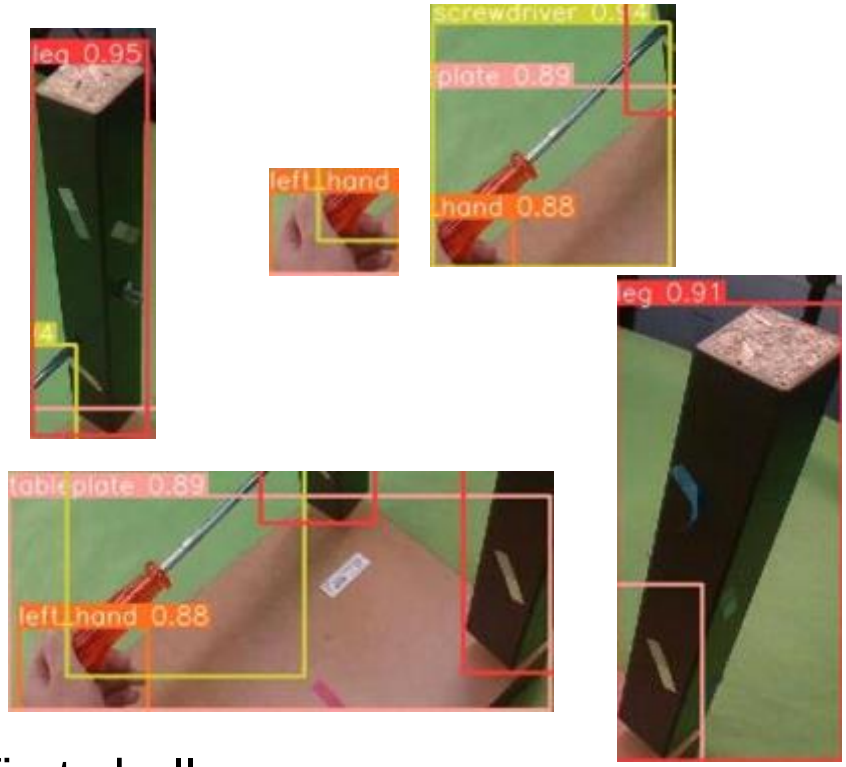
Activity Recognition - Challenges

- two major challenges:
  - good part of frames look similar

  - unique identification possible?

Activity Recognition - Ideas

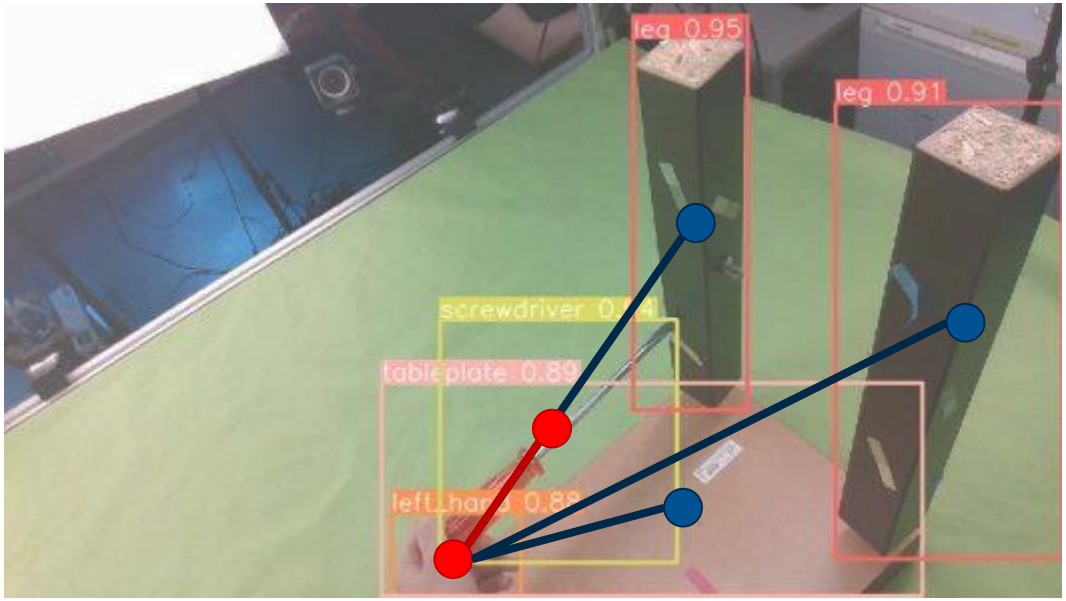
- instead of using the whole image → Region of Interest (ROI)



- this solves the first challenge

Activity Recognition - Ideas

- use distances as additional information



center of bounding box …

- of interest
- not of interest

- calculate Euclidean Distance between objects: $d(q,p) = \sqrt{(p-q)^2}$
- idea: in general, hand is closer to the object defining the activity

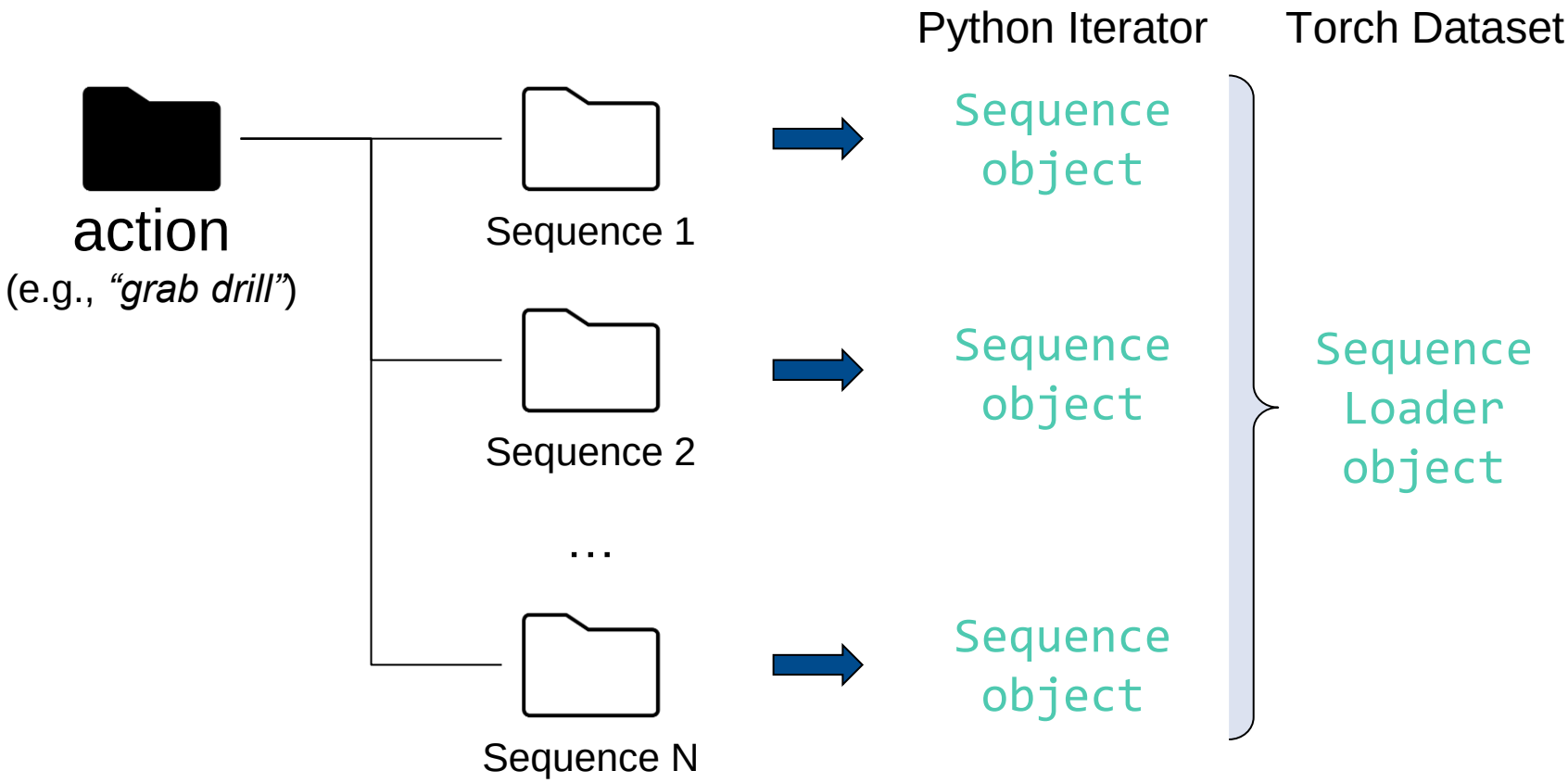
Activity Recognition - Ideas

- instead of using frames alone, we use sequences

- this solves the second challenge

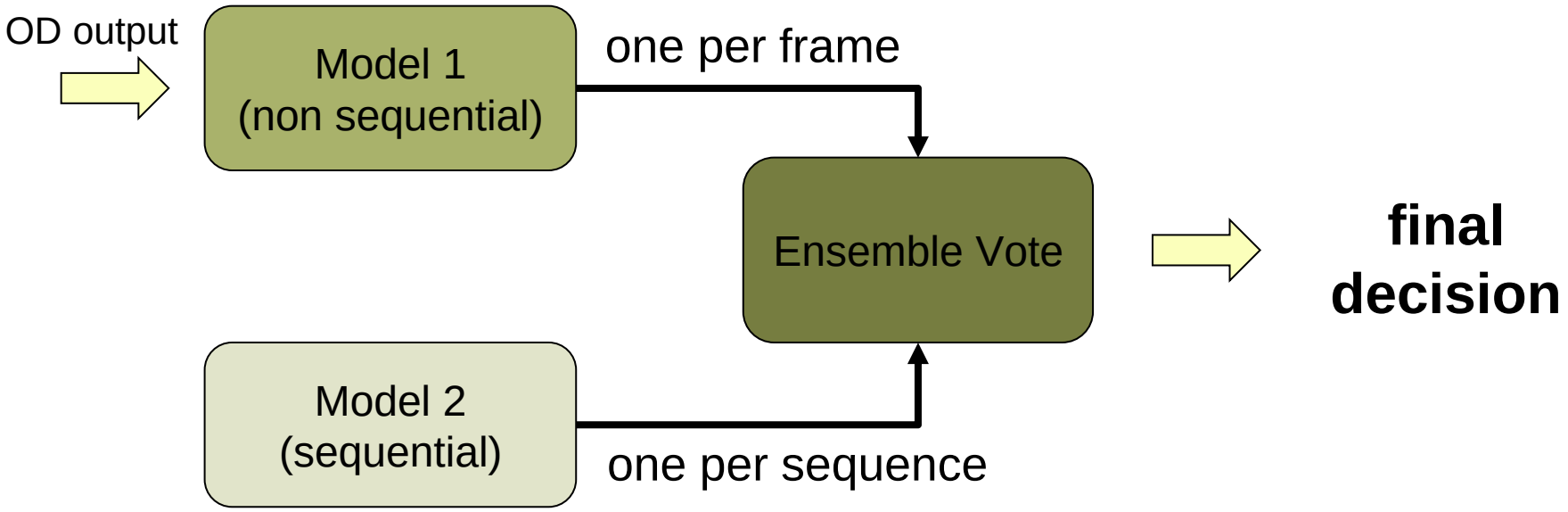
frames from previous slides

Activity Recognition - Annotation & Data loading



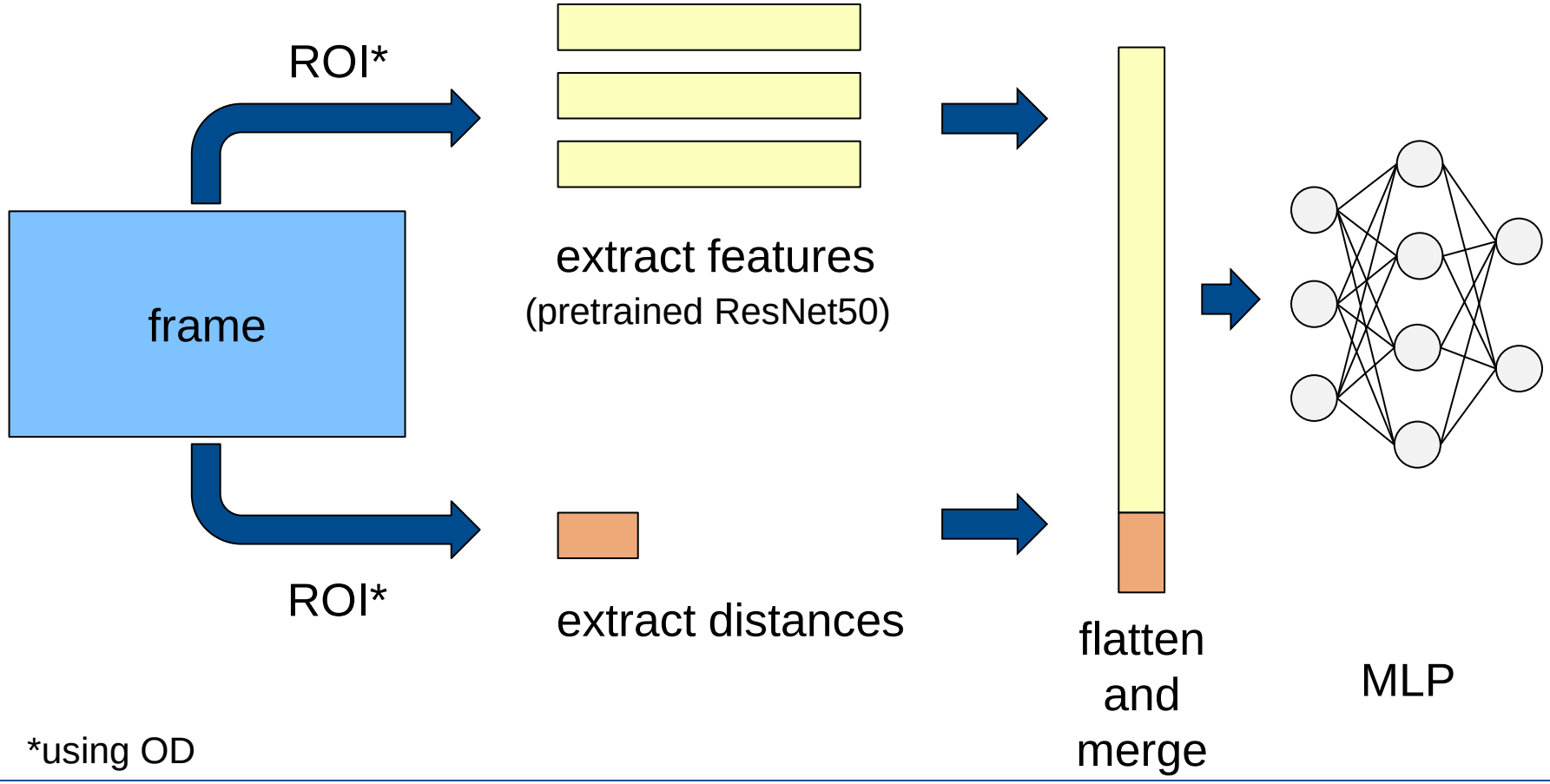
Activity Recognition - Architecture

- to incorporate both ideas in the architecture, I built two models:



Activity Recognition - Architecture

- Model 1:



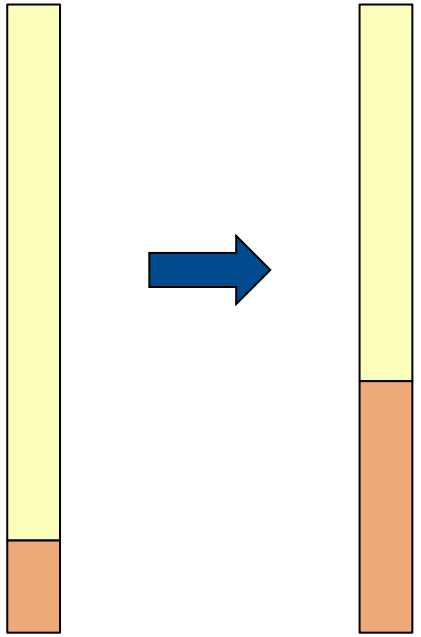
*using OD

Activity Recognition - Architecture

- features have higher dimension → distance vector is "overshadowed" during learning process
- two possible solutions:
  - reduce feature dimension
  - introduce weights for distance vector

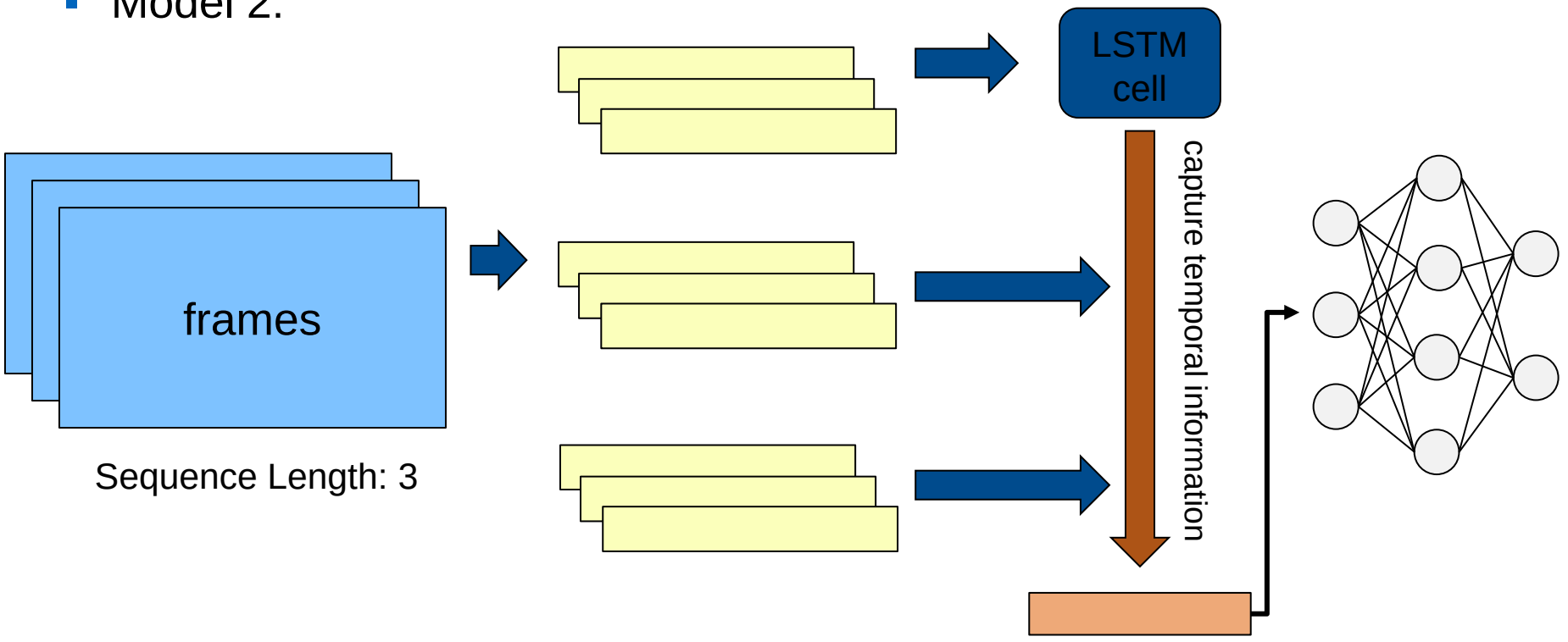
sum of feature vectors

sum of distance vectors



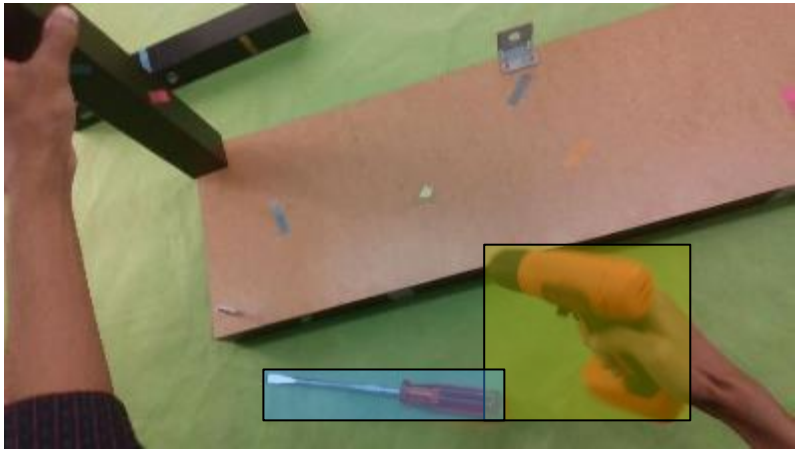
Activity Recognition - Architecture

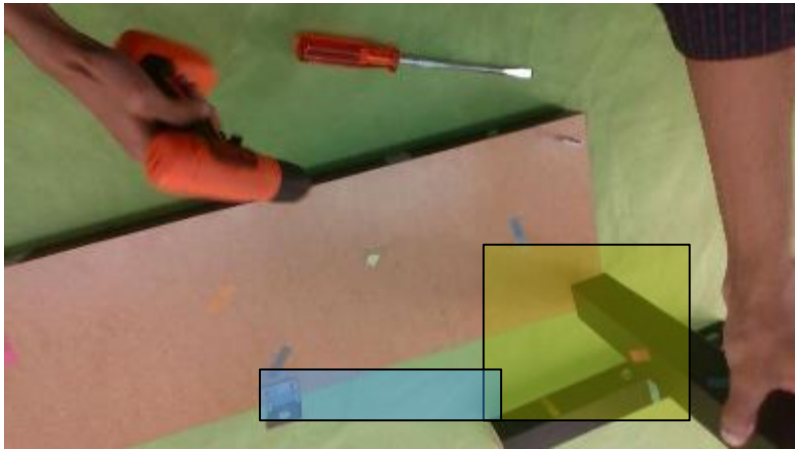
- Model 2:



Activity Recognition - Augmentation

- helps enriching the dataset
- (in general) leads to better robustness
- challenge: bounding boxes must be transformed as well



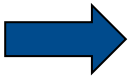


Activity Recognition - Evaluation

- hyperparameter tuning using **Ray Tune**

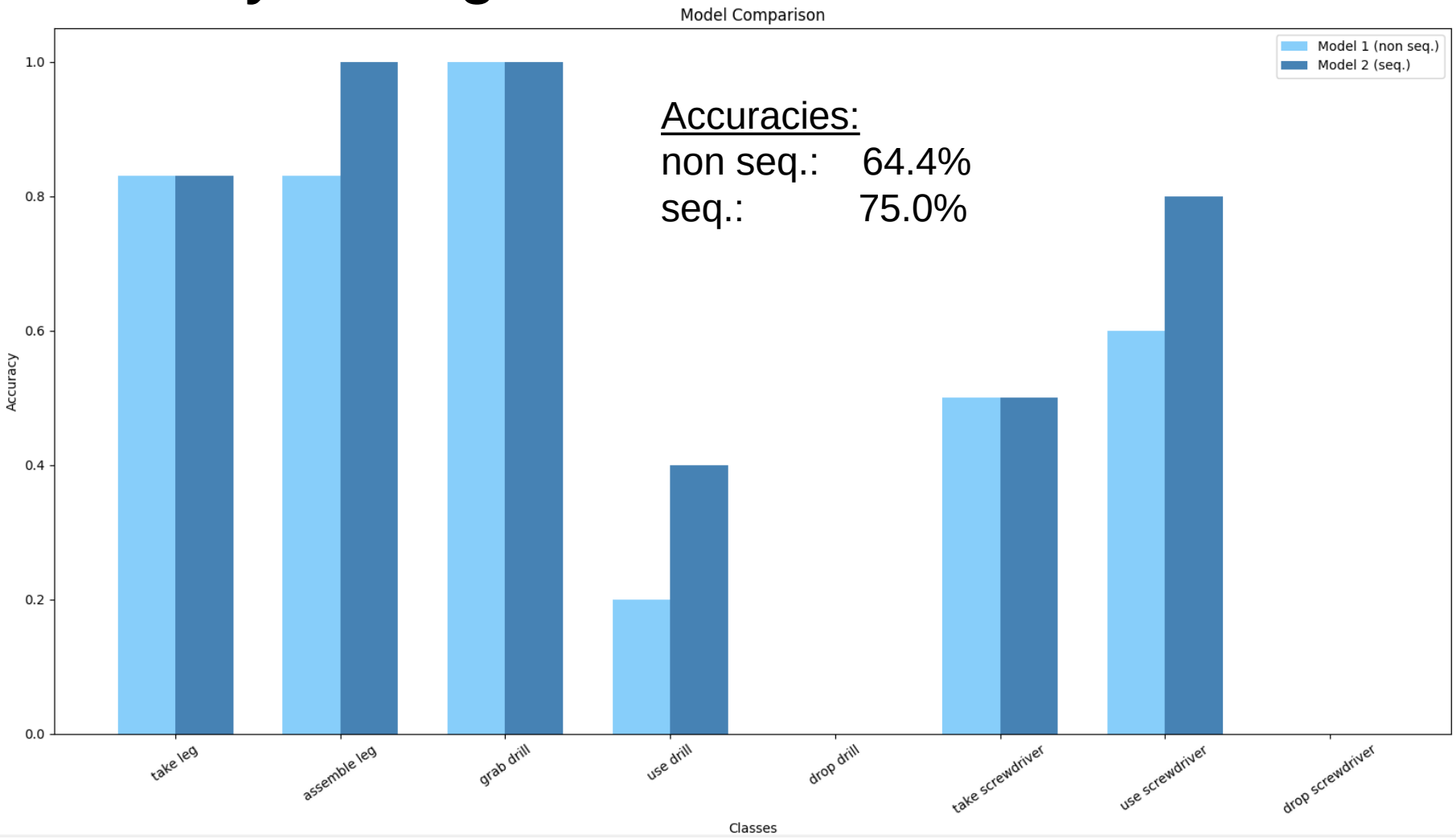
Trial name	status	loc	lr	fc1	fc2	feature_size	weight	drop_rate
training_822e1_00000	RUNNING	[redacted]	0.001	512	256	1000	1	0.1
training_822e1_00001	PENDING		0.001	2048	1024	1000	3	0.3
training_822e1_00002	PENDING		0.001	256	128	100	1	0.1
training_822e1_00003	PENDING		0.001	512	256	100	3	0.3
training_822e1_00004	PENDING		0.01	1024	512	1000	1	0.1
training_822e1_00005	PENDING		0.01	512	256	100	3	0.1
training_822e1_00006	PENDING		0.1	1024	512	1000	1	0.1
training_822e1_00007	PENDING		0.1	512	256	100	3	0.1

- complete pipeline accuracy: **53.6%** (46.4% without landmark detection)
- activity recognition only accuracy: **64.4%**

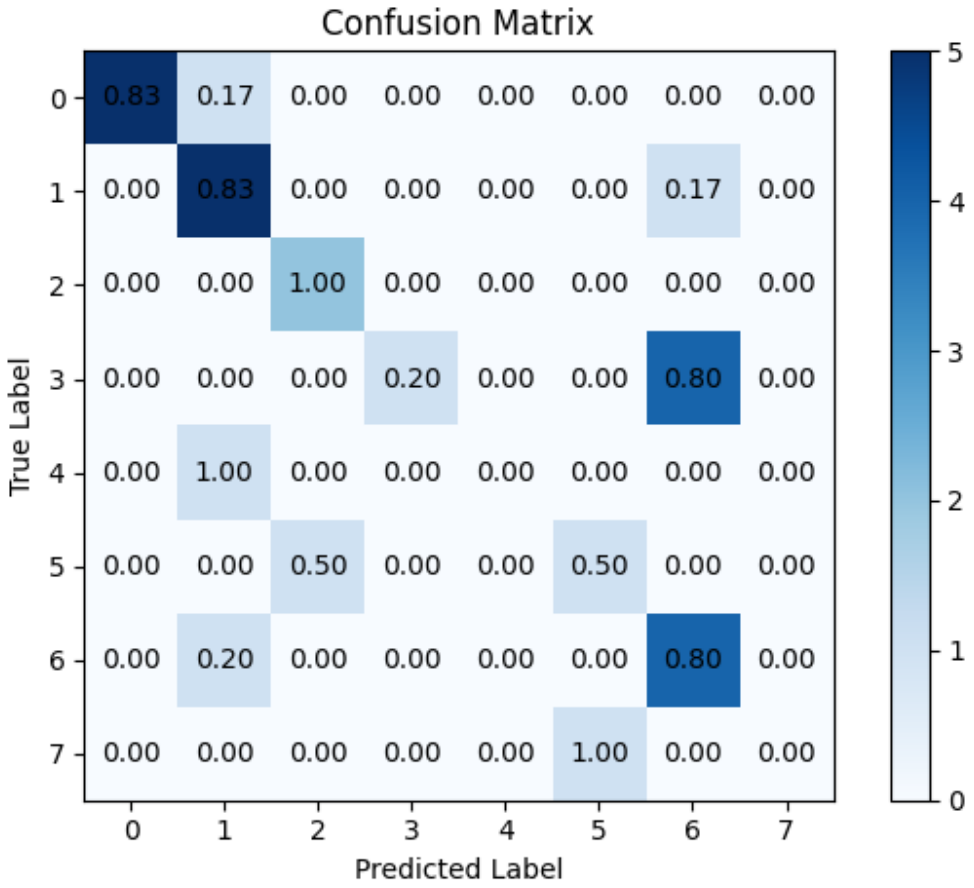


large error propagation

Activity Recognition - Evaluation



Activity Recognition - Evaluation

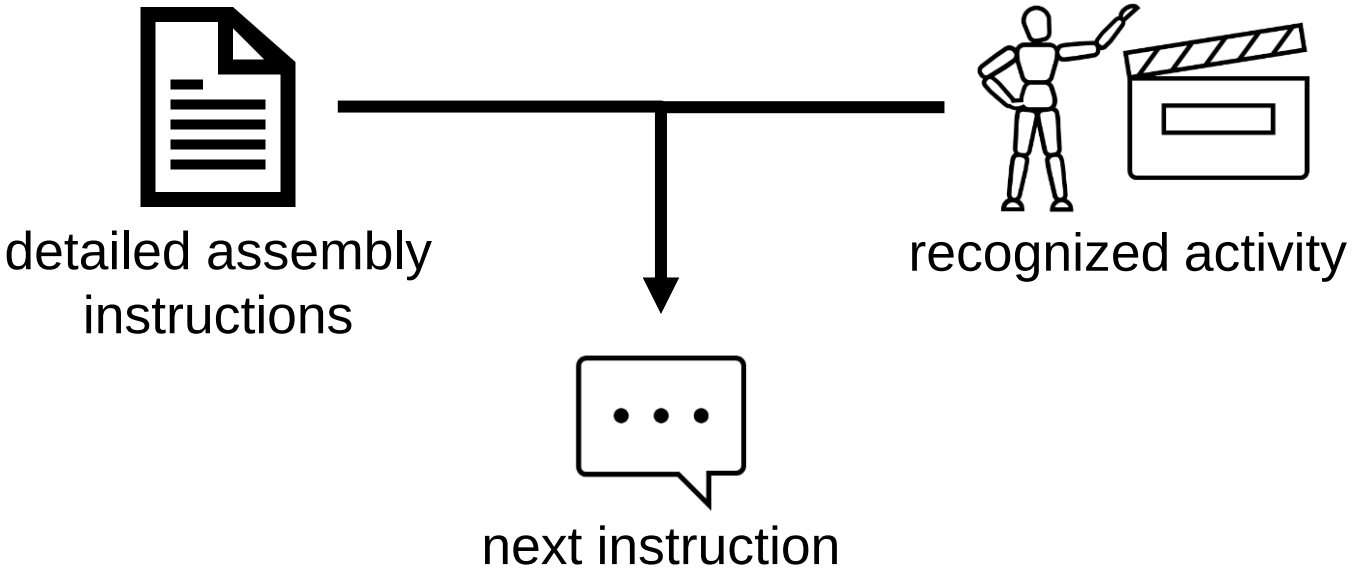


class	label
0	take leg
1	assemble leg
2	grab drill
3	use drill
4	drop drill
5	take screwdriver
6	use screwdriver
7	drop screwdriver

LLM Integration - Process

LangChain (4)

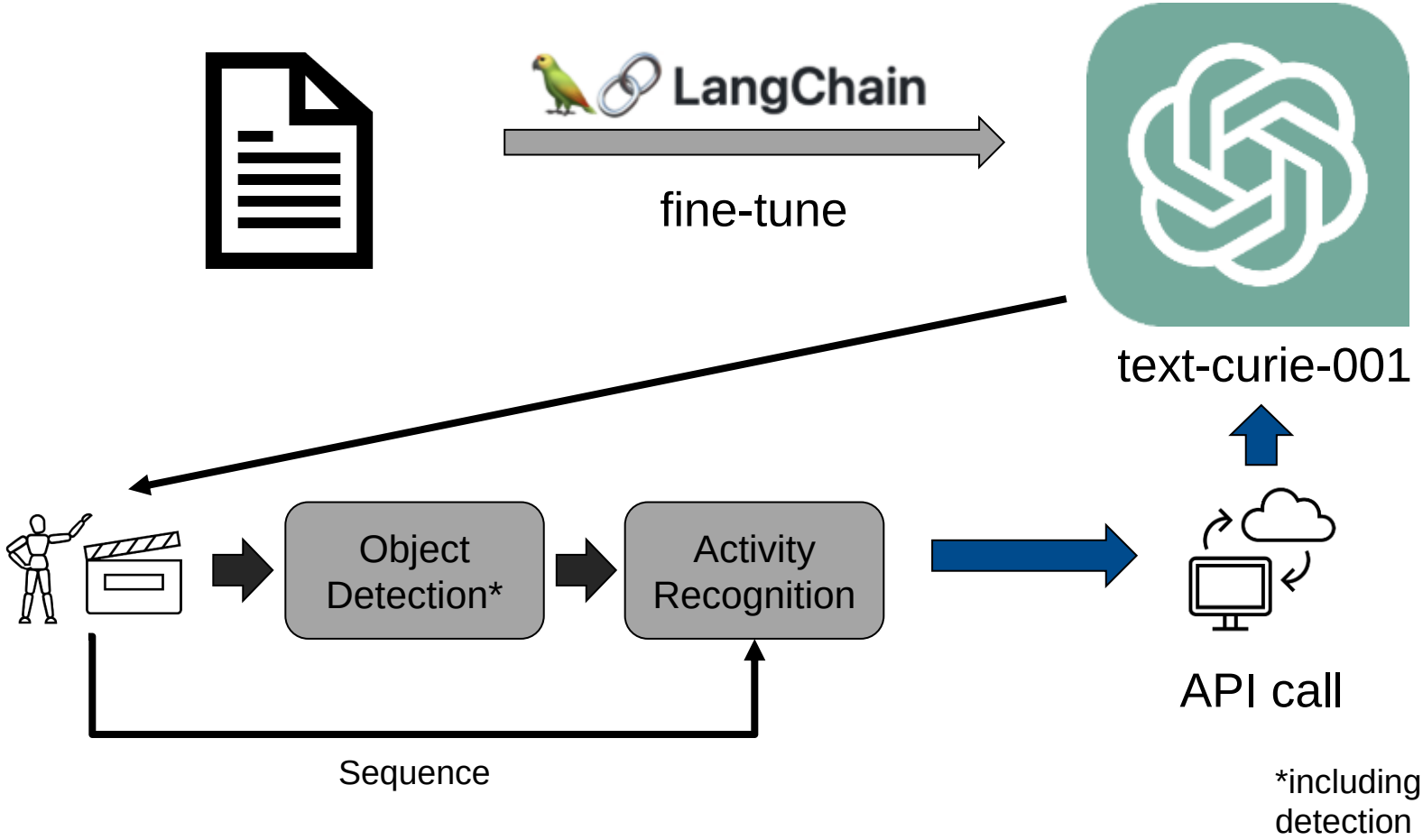
- remember: our goal is to have a “Jarvis-like” assistant
- LangChain:
  - MIT-licensed framework to streamline creation LLM applications
  - used to build chatbots for example



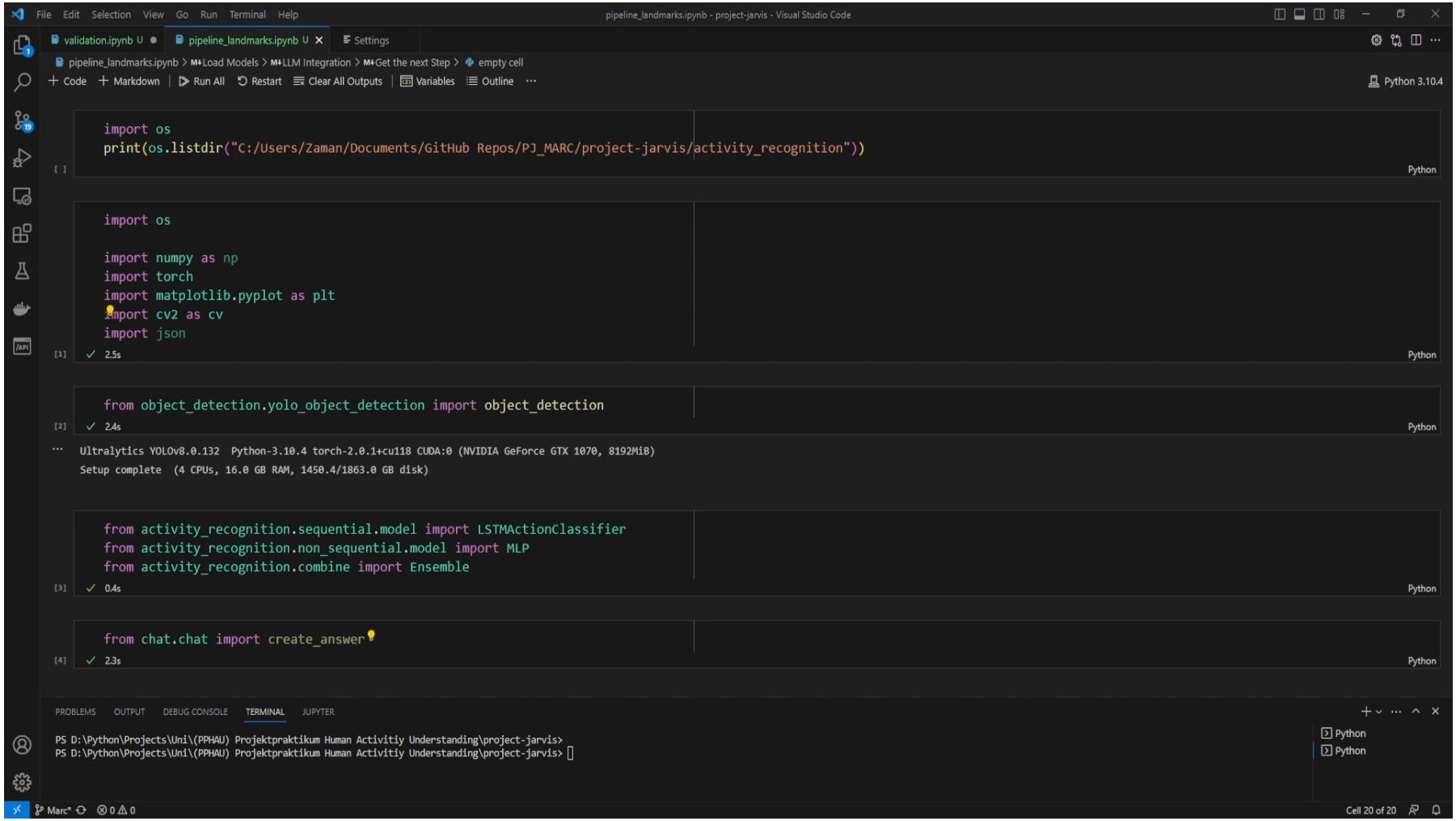
LLM Integration - Process

(5)

LLM Integration – Process (detailed)



Demo


File Edit Selection View Go Run Terminal Help
pipeline_landmarks.ipynb - project-jarvis - Visual Studio Code
validation.ipynb U
pipeline_landmarks.ipynb U
Settings
pipeline_landmarks.ipynb > Load Models > LLM Integration > Get the next Step > empty cell
+ Code + Markdown Run All Restart Clear All Outputs Variables Outline
Python 3.10.4
import os
print(os.listdir("C:/Users/Zaman/Documents/GitHub Repos/PJ_MARC/project-jarvis/activity_recognition"))
Python
import os
import numpy as np
import torch
import matplotlib.pyplot as plt
import cv2 as cv
import json
2.5s
Python
from object_detection.yolo_object_detection import object_detection
2.4s
Python
Ultralytics YOLOv8.0.132 Python-3.10.4 torch-2.0.1+cu118 CUDA:0 (NVIDIA GeForce GTX 1070, 8192MiB)
Setup complete (4 CPUs, 16.0 GB RAM, 1450.4/1863.0 GB disk)
from activity_recognition.sequential.model import LSTMActionClassifier
from activity_recognition.non_sequential.model import MLP
from activity_recognition.combine import Ensemble
0.4s
Python
from chat.chat import create_answer
2.3s
Python
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
PS D:\Python\Projects\Uni\(PPHAU) Projektpraktikum Human Activitiy Understanding\project-jarvis>
PS D:\Python\Projects\Uni\(PPHAU) Projektpraktikum Human Activitiy Understanding\project-jarvis>
Python
Python
Marc*
Cell 20 of 20

Activity Recognition - Transformer

Muhammad Hamas Khan

Activity Recognition - Challenges

- Small Dataset – Not Diverse
- Spatio-Temporal Awareness – Models like MLP are a feedforward neural network without spatio-temporal awareness.

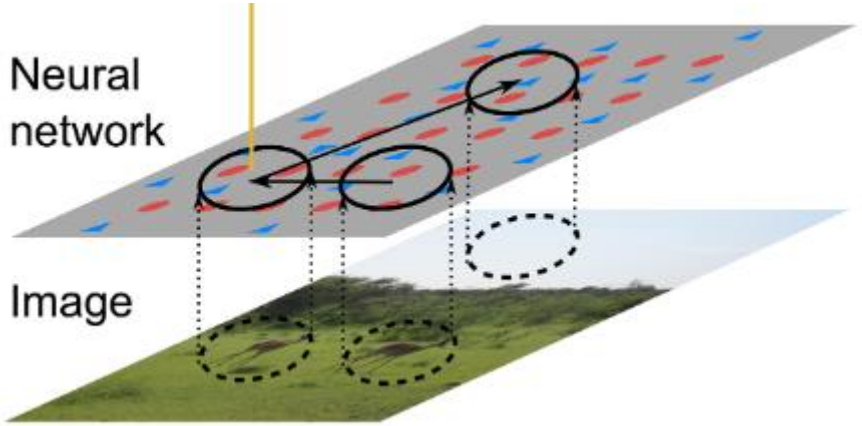
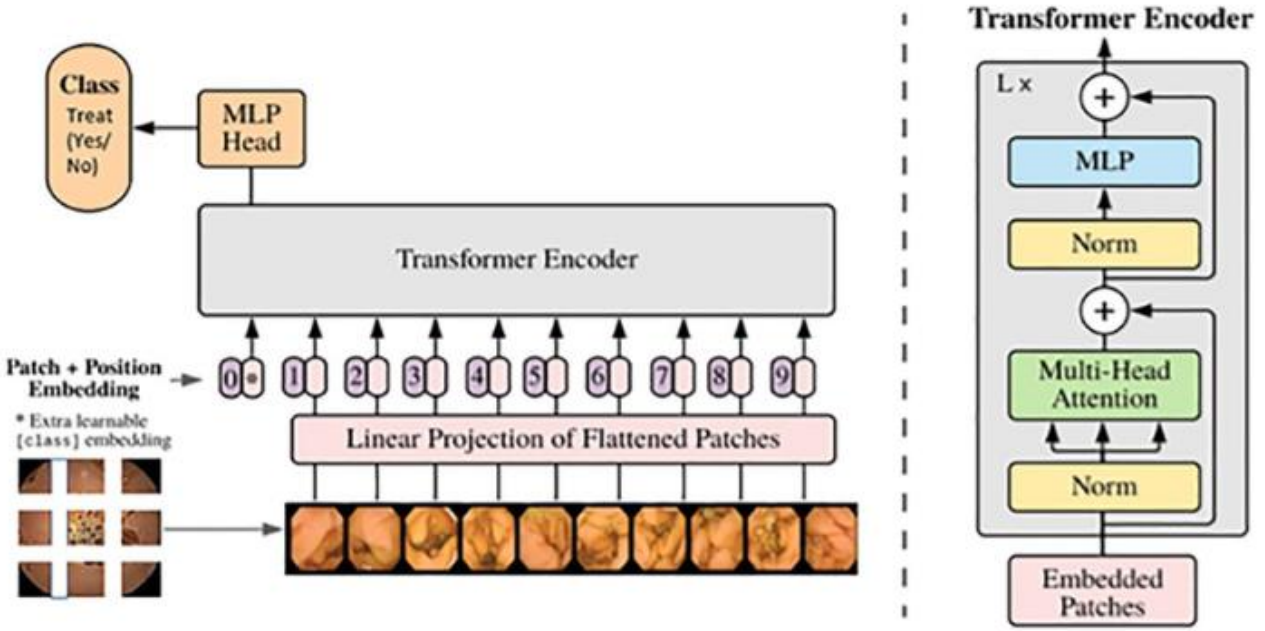


Image reference in slide notes

Activity Recognition - Challenges

Possible Solution?

Activity Recognition – Transformers Architecture



An overview of the TimeSformer architecture. On the right side: the video self-attention block, applied on the embedded patches. On the left side: the end-to-end architecture. A multilayer perceptron (MLP) is applied both at the end of each block and to the projected and concatenated vectors from all heads.

Transformers - Advantages

- **Attention mechanism**:
  - Activity recognition often requires capturing dependencies between actions occurring at different time steps.
  - Excels at modeling long-range dependencies
  - Individual actions like pouring milk vs classifying the complex activity (many recipes involve pouring milk).

Transformers - Advantages

- **Parallel Computation**:
  - Processes the entire input sequence in parallel.
  - Enables faster training and inference
  - Beneficial when dealing with long videos or time series data.

Transformers - Advantages

- **Global context modeling**:
  - Transformers capture global dependencies by attending to all positions in the sequence simultaneously.
  - Allowing the model to consider the entire context of an activity, including actions that occur earlier or later in the sequence.

Activity Recognition – Modeling Challenge

- **Computational Complexity:**

The Meta's model was trained using four GPU compute nodes, each containing 8 Tesla V100 GPUs (32 GPUs total).

- **Insufficient Data:**

Data is not diverse enough to train such a Complex model.

Meta's Model DataSet
136M video clips with captions sourced
from 1.2M Youtube videos (15 years of video) 23k activities

Activity Recognition – Transfer Learning

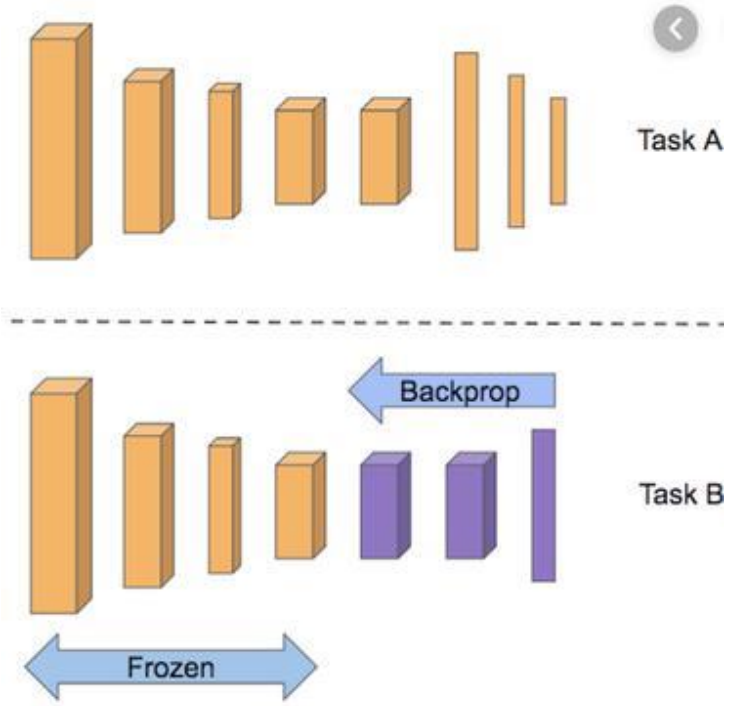
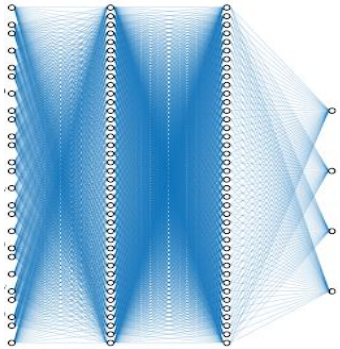


Image reference in slide notes

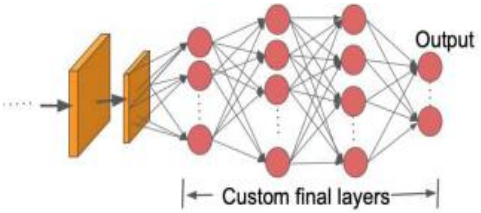
Activity Recognition – Architecture Adaptation

Pretrained Weights on Kinetics 400 Dataset



Features Extracted

Custom Classification



Tuning and Optimization

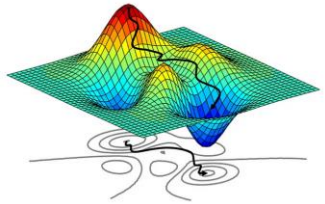
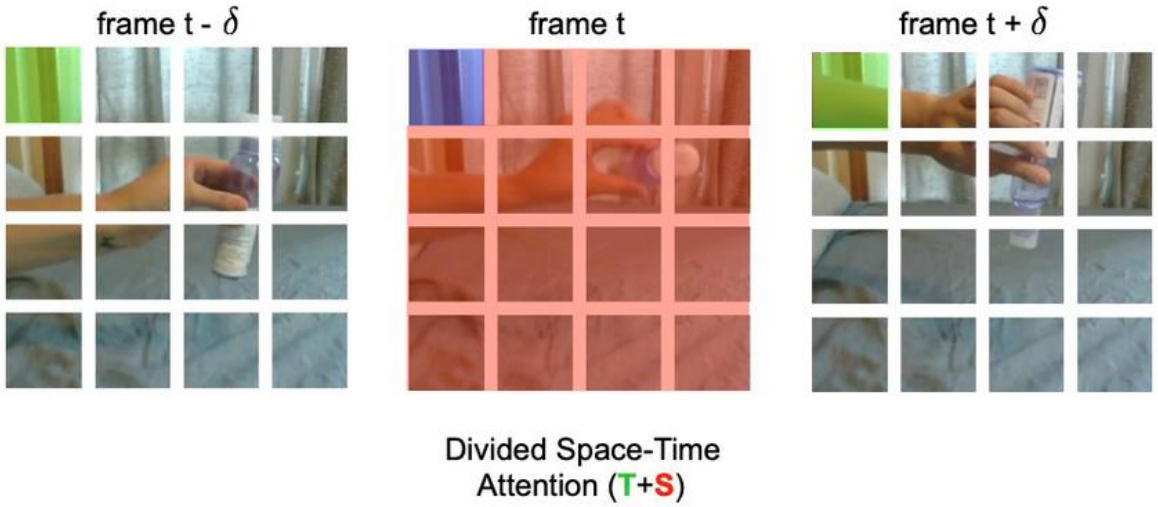


Image references in slide notes

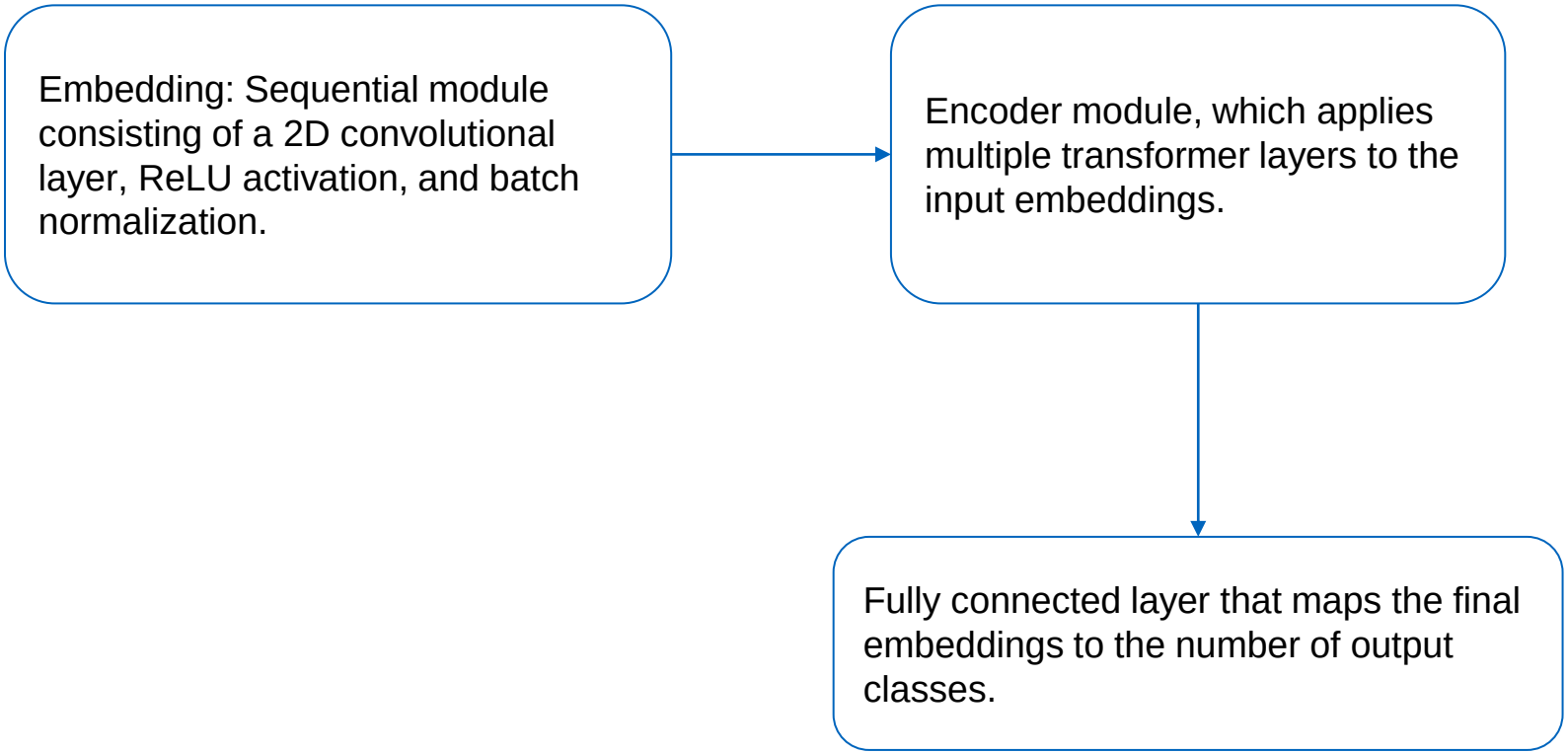
Activity Recognition – TimesFormer Backbone



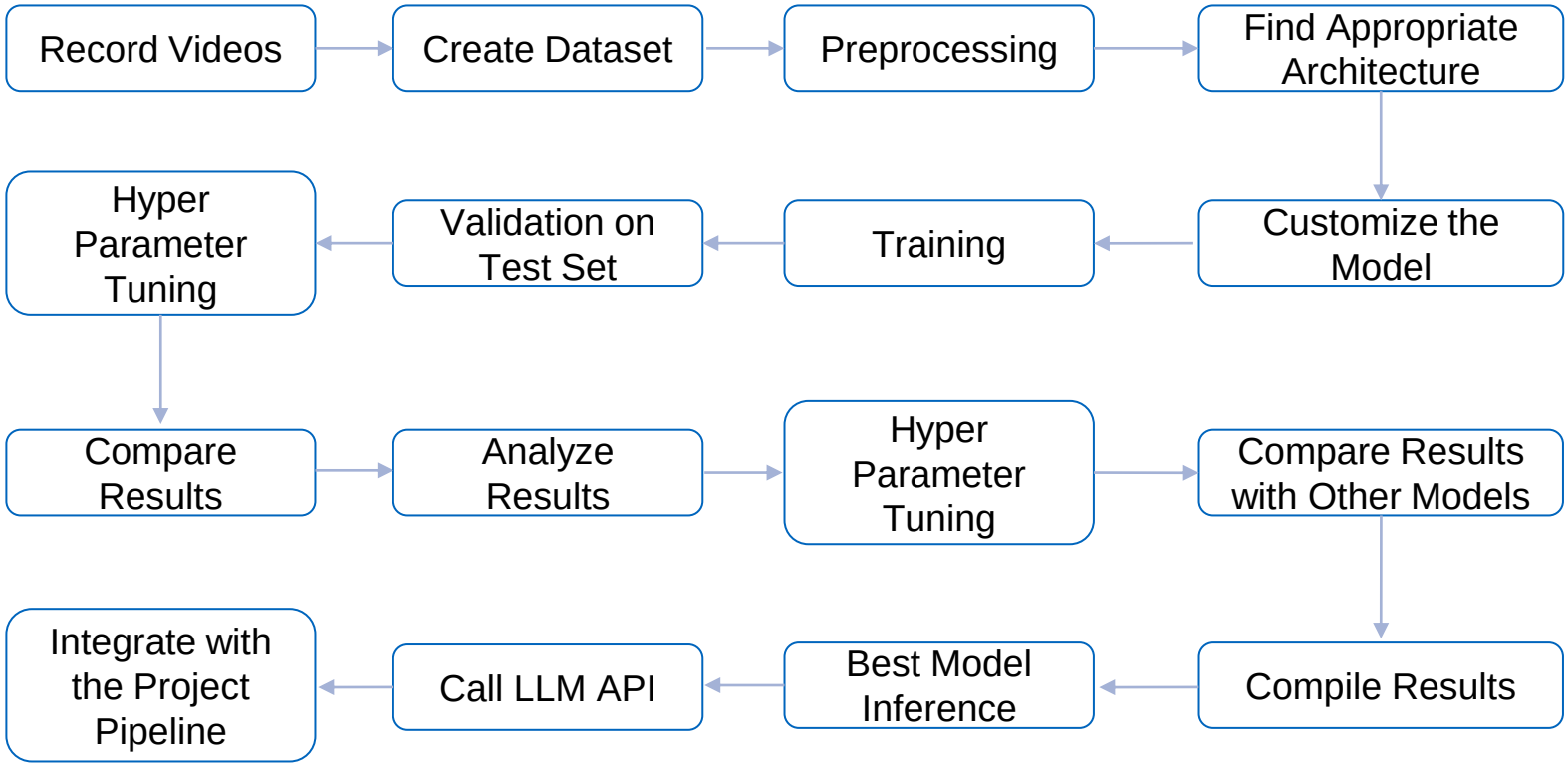
- T temporal comparisons are made for each patch. (green patches)
- Spatial attention: Patch is compared only with patches N within the same frame (red patches).
- Divided space-time attention performs in total only (T+N) comparison per patch rather than T*N.

Reference: https://ai.facebook.com/blog/timesformer-a-new-architecture-for-video-understanding/

Activity Recognition – Transformer Modules



Transformer Network – Steps Taken and Flow



Activity Recognition – Data Sets

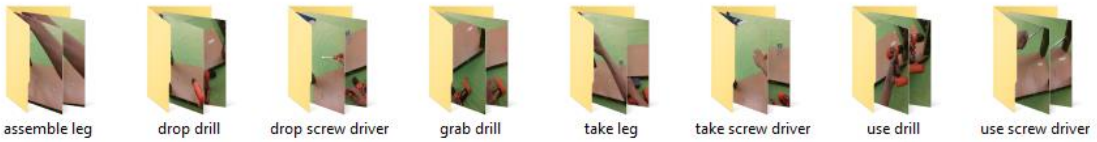
Training
296 Images
(80-20 Split)

Validation
74 Images
(80-20 Split)

Test
96 Images
(Avg. 10- 15 per Class)

Sequence Length
Original image size = 224x224
Patch_size = 32

Image divided into 7x7 patches, resulting in a sequence length of 49 (7x7)



Activity Recognition – Our Architecture

```python
class Transformer(nn.Module):
    def __init__(self, num_classes, embed_dim, num_heads, num_layers, patch_size, dropout_pro
        super(Transformer, self).__init__()

        self.embed_dim = embed_dim
        self.patch_size = patch_size

        self.embedding = nn.Sequential(
            nn.Conv2d(3, embed_dim, kernel_size=patch_size, stride=patch_
            nn.ReLU(),
            nn.BatchNorm2d(embed_dim)
        )

        self.transformer = nn.TransformerEncoder(
            nn.TransformerEncoderLayer(d_model=embed_dim, nhead=num_heads
            num_layers=num_layers
        )

        self.fc = nn.Linear(embed_dim, num_classes)
        self.dropout = nn.Dropout(p=dropout_prob)

    def forward(self, x):
        batch_size, _, _, _ = x.shape

        x = self.embedding(x)
        x = x.permute(2, 3, 0, 1)

        seq_length = x.shape[2]
        x = x.reshape(-1, seq_length, self.embed_dim)

        x = self.transformer(x)
```

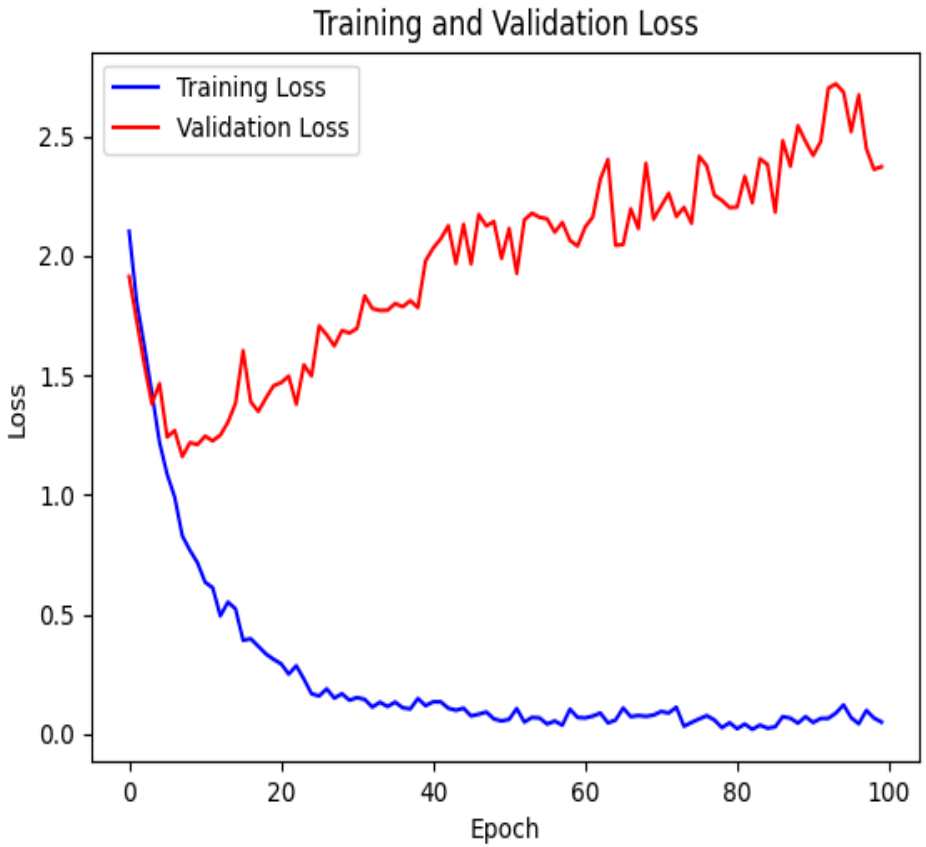
```python
for epoch in range(num_epochs):
    # Training Phase
    pretrained_model.train()  # Set the model to training mode
    train_epoch_loss = 0.0

    for i, (frames, labels) in enumerate(train_dataloader):
        frames = frames.to(device)
        labels = labels.to(device)

        # This is the Forward Pass on the model
        outputs = pretrained_model(frames)

        # Computing the loss between model inference and target labels
        loss = criterion(outputs, labels)

        # Regularization - L2 regularization for all trainable parameters
        l2_reg = torch.tensor(0.).to(device)
        for param in pretrained_model.parameters():
            l2_reg += torch.norm(param)

        loss += l2_lambda * l2_reg

        # Backward pass/propagation and optimization function call
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        train_epoch_loss += loss.item()

        if (i + 1) % 10 == 0:
            print(f"Epoch [{epoch+1}/{num_epochs}], Train Step [{i+1}/{len(train_dataloader)}]
```

Transformer Training – HyperParameters

```
# Hyperparameters
num_classes = 8
patch_size = 16
embed_dim = 256
num_heads = 4
num_layers = 2
batch_size = 4
learning_rate = 1e-5
num_epochs = 50
dropout_prob = 0.5
l2_lambda = 1e-5
```

```
num_classes = 8
patch_size = 32
embed_dim = 256
num_heads = 2
num_layers = 1
batch_size = 8
learning_rate = 1e-4
num_epochs = 100
dropout_prob = 0.5
l2_lambda = 1e-5
```

Transformer Training – Overfitting Issue



Poor performance

Overfitting Issue – Techniques Used to Resolve

1. Reducing model complexity
2. Regularization
3. Dropout
4. Learning rate scheduling
5. Data Augmentation (Random Rotation, Random Horizontal Flip)

Resolving Overfitting Issue – Regularization

1. Used L2 Regularization

1. Prevents overfitting by discouraging the model from relying too much on any single feature.

```
l2_lambda = 1e-5
```

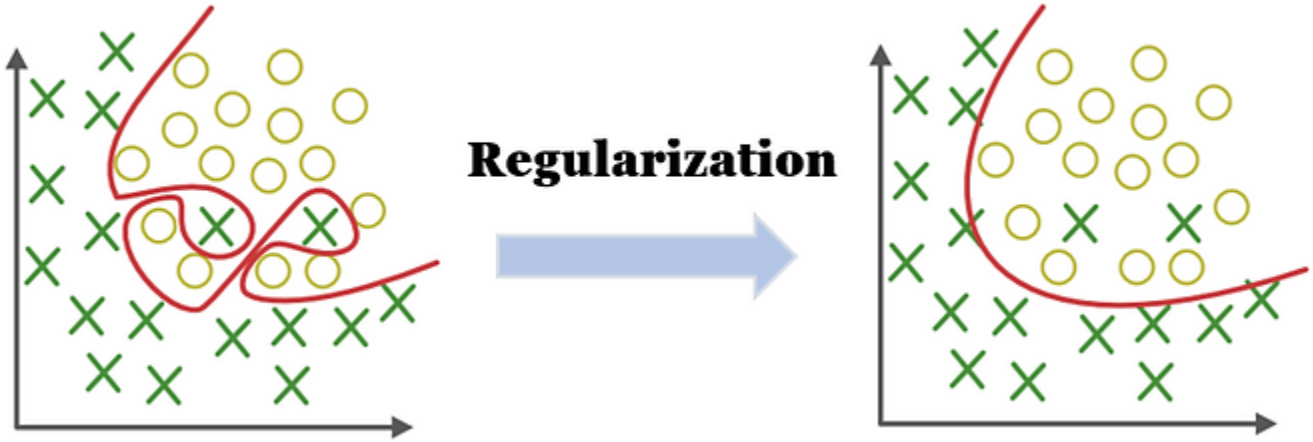


Image reference in slide notes

Resolving Overfitting Issue – Dropout

1. Randomly set to zero a fraction of neurons

1. Effectively creates an ensemble of multiple smaller networks, as it trains on different subsets of the neurons during each iteration.

```
dropout_prob = 0.5
```

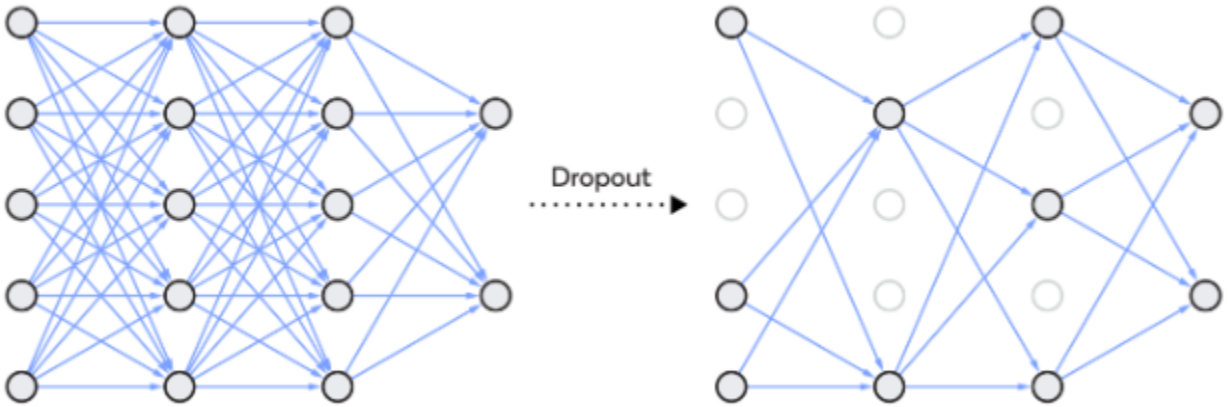


Image reference in slide notes

Resolving Overfitting Issue – Learning Rate Scheduling

1. Dynamically change the learning rate during training
1. Achieve better convergence
1. Prevents overshooting or getting stuck in local minima.

```
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=20, gamma=0.5)
```

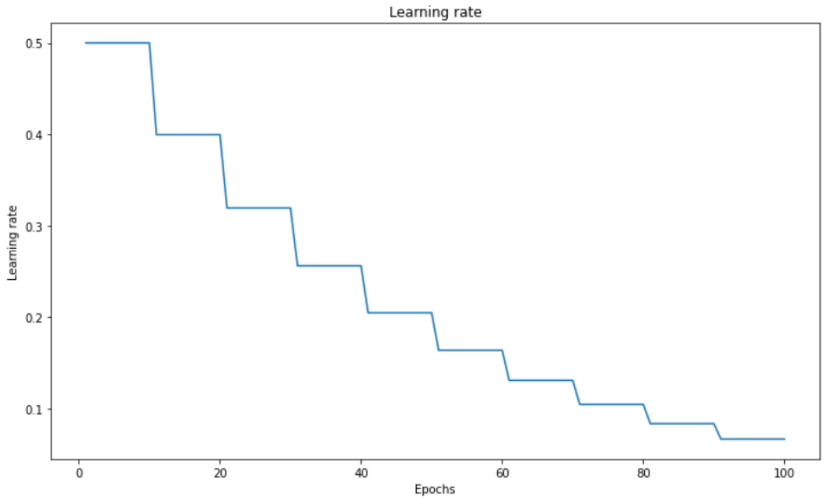
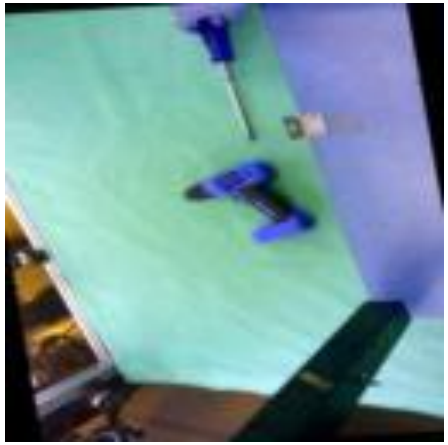
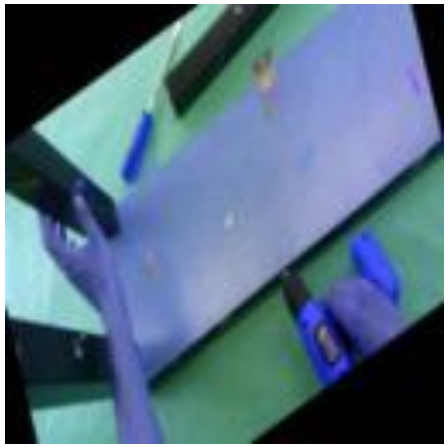


Image reference in slide notes

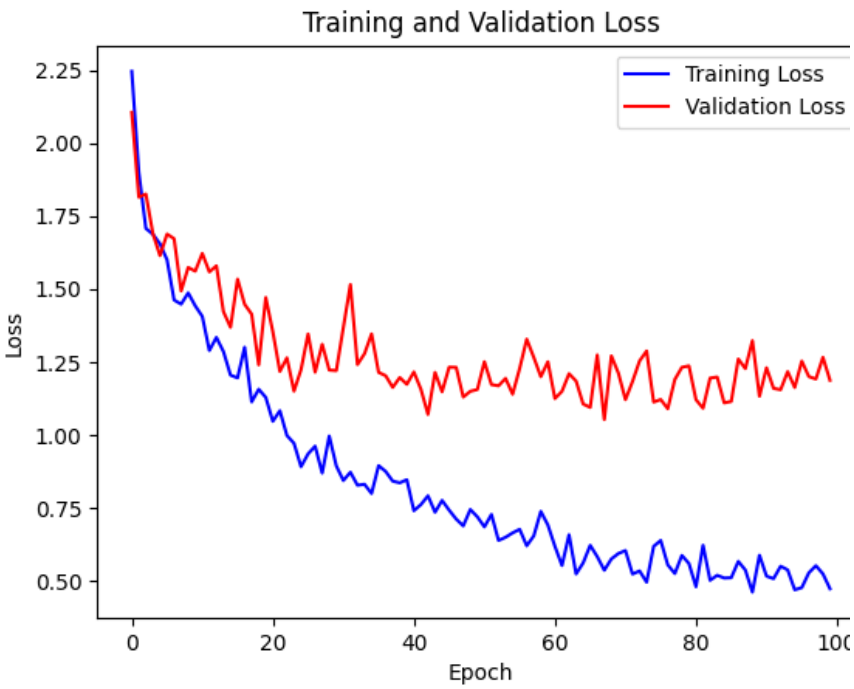
Resolving Overfitting Issue – Data Augmentation

```
# Transformation Pipeline
transform = Compose([
    Resize((patch_size, patch_size)),
    ToTensor(),
    torchvision.transforms.RandomRotation(10),
    torchvision.transforms.RandomHorizontalFlip(p=0.5),
])
```

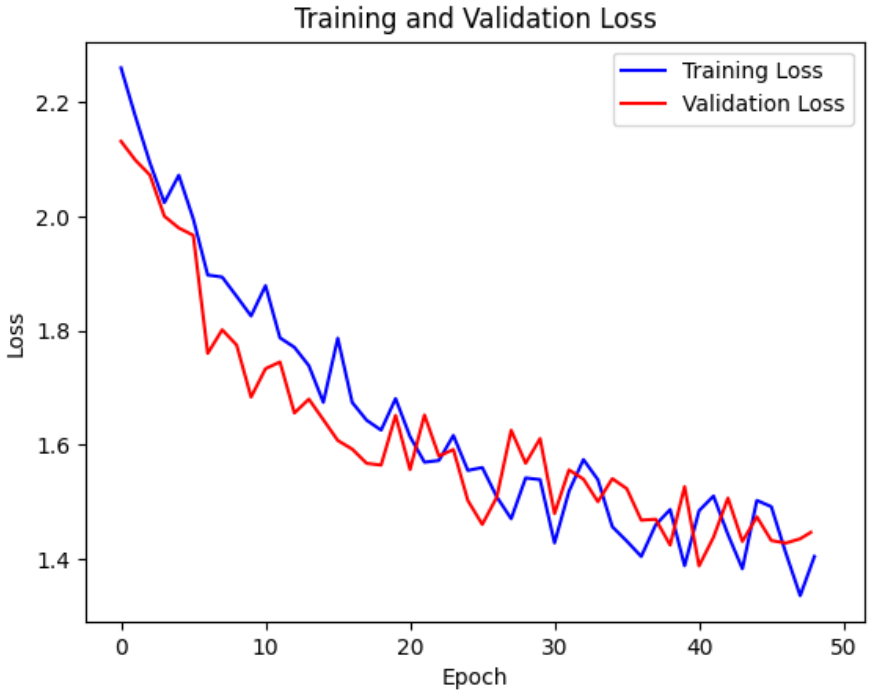




Resolving Overfitting Issue – Training Validation



Average Performing Model



Best Performing Model

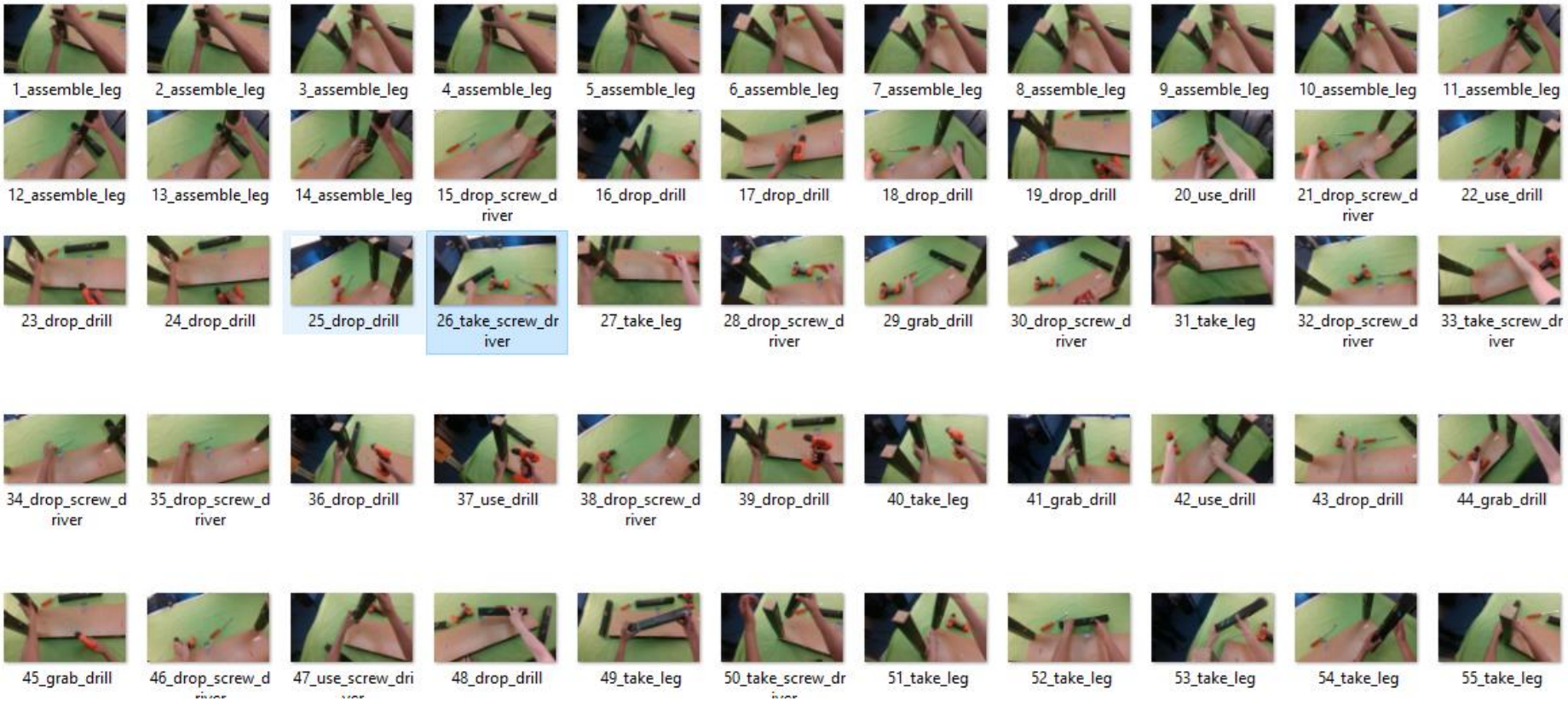
Transformer – Model Evaluation

Parameters	Model 1	Model 2	Model 3
Patch Size	16	32	32
Embedding Dimension	128	256	256
Number Heads	4	4	2
Number Layers	2	2	1
Batch Size	2	4	4
Learning Rate	1 e-4	Adaptive	Adaptive
Number Epochs	50	100	100
Training – 80 %	296	296	296
Validation – 20 %	74	74	74
Dropout Prob.	None	0.4	0.5
L2 Regularization	None	1 e-5	1 e-5
Test Correct Predictions	**53 %**	**66.6 %**	**74 %**

Transformer – Action Classification Analysis

	Training-Validation	Test	Worst Model	Avg. Model	Best Model
Assemble Leg	50	15	11	14	14
Drop Drill	40	10	4	4	7
Drop Screw Driver	37	10	3	5	6
Grab Drill	45	11	2	2	5
Take Leg	50	15	9	12	12
Take Screw Driver	48	10	5	5	5
Use Drill	50	10	6	8	9
Use Screw Driver	50	15	11	14	13
Total	370	96	51	64	71
% Correct	-	-	53 %	66.6 %	74 %

Transformer – Output



Thank You

References

(1) https://upload.wikimedia.org/wikipedia/commons/thumb/0/04/ChatGPT_logo.svg/1200px-ChatGPT_logo.svg.png
(2) https://www.cinepremiere.com.mx/wp-content/uploads/2020/08/Iron-Man-JARVIS.jpg
(3) https://miro.medium.com/max/10240/1*9sUmXnMquEokqPnHLeqDmg.jpeg
(4) https://assets.st-note.com/production/uploads/images/94297454/rectangle_large_type_2_91064e9f21492d5520c7b612794783c4.png?width=800
(5) https://www.ikea.com/gb/en/images/products/lack-side-table-black__22518_pe107397_s5.jpg
(6) https://images2.minutemediacdn.com/image/upload/c_fill,g_auto,h_1248,w_2220/v1555922266/shape/mentalfloss/screen_shot_2014-08-18_at_2.45.22_pm.png?itok=7vD5puwo

References

(7) https://docs.ultralytics.com/
(8) https://docs.ray.io/en/latest/tune/index.html
(9) https://github.com/ultralytics/ultralytics
(10) https://arxiv.org/pdf/2102.05095.pdf
(11) https://ai.facebook.com/blog/timesformer-a-new-architecture-for-video-understanding/
(12) https://github.com/facebookresearch/TimeSformer
(13) arxiv.org/abs/1506.02640