Precision at Pixel-Level: YOLOv8 vs. Conventional UI Testing

© Nikolaus Rieder

FH University of Applied Sciences TECHNIKUM WIEN

About me

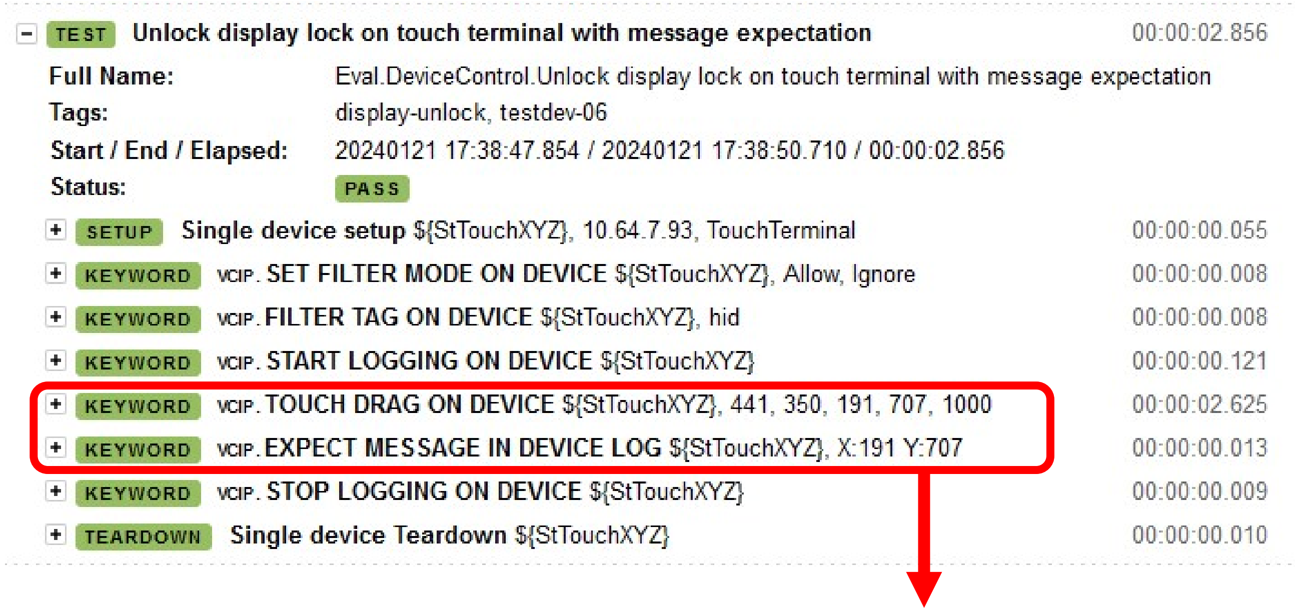
- Test automation engineer
- Working in quality assurance for 7 years
  - Experience in manual & automation testing
- Testing nurse call systems

SCHRACK
SECONET

Before we dive in…

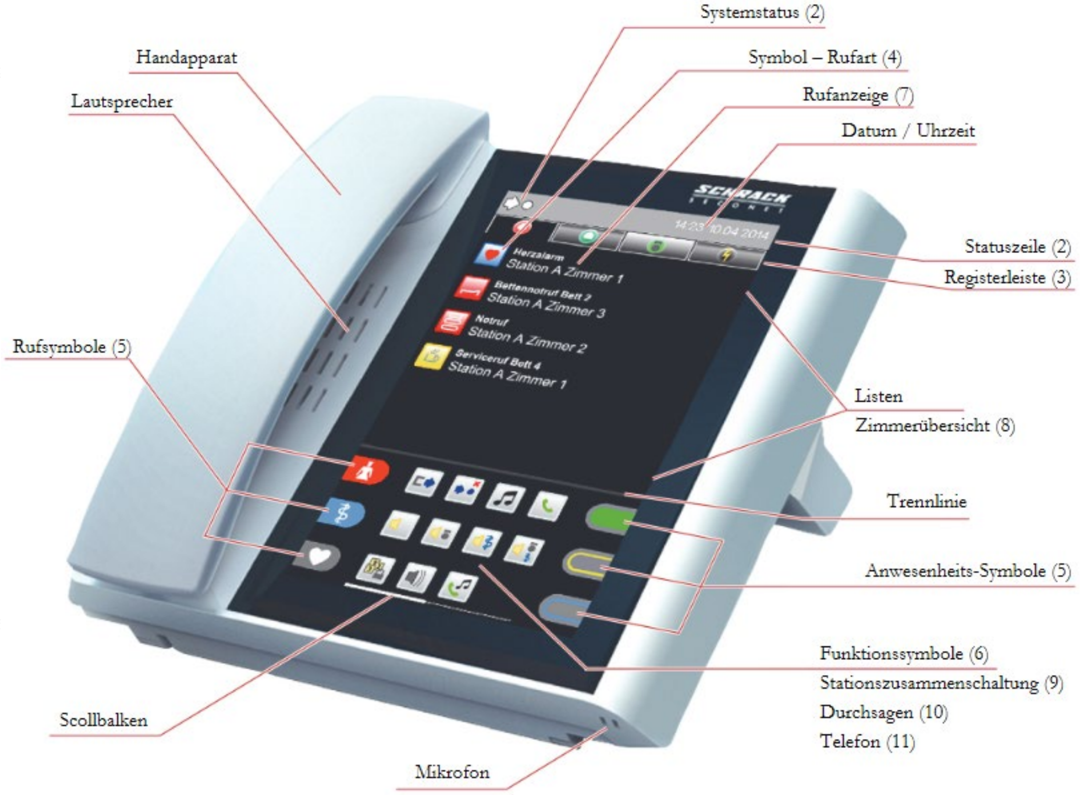
- Manual UI testing is labor-intensive
- Testing embedded devices demands extra effort
- More & better automation equals reduced workload
- But…
  - very error-prone
  - High maintenance
  - Difficult to simulate user input
  - UI change? → Rework test cases

What does UI test automation look like?



```
− TEST  Unlock display lock on touch terminal with message expectation                00:00:02.856
   Full Name:            Eval.DeviceControl.Unlock display lock on touch terminal with message expectation
   Tags:                 display-unlock, testdev-06
   Start / End / Elapsed: 20240121 17:38:47.854 / 20240121 17:38:50.710 / 00:00:02.856
   Status:               PASS
   + SETUP     Single device setup ${StTouchXYZ}, 10.64.7.93, TouchTerminal                 00:00:00.055
   + KEYWORD   vcip. SET FILTER MODE ON DEVICE ${StTouchXYZ}, Allow, Ignore                00:00:00.008
   + KEYWORD   vcip. FILTER TAG ON DEVICE ${StTouchXYZ}, hid                               00:00:00.008
   + KEYWORD   vcip. START LOGGING ON DEVICE ${StTouchXYZ}                                 00:00:00.121
   + KEYWORD   vcip. TOUCH DRAG ON DEVICE ${StTouchXYZ}, 441, 350, 191, 707, 1000          00:00:02.625
   + KEYWORD   vcip. EXPECT MESSAGE IN DEVICE LOG ${StTouchXYZ}, X:191 Y:707               00:00:00.013
   + KEYWORD   vcip. STOP LOGGING ON DEVICE ${StTouchXYZ}                                  00:00:00.009
   + TEARDOWN  Single device Teardown ${StTouchXYZ}                                        00:00:00.010
```

UI interactions require exact coordinates to simulate user input



Goals

- Automate widget detection in UI screenshots
- Boost test accuracy and development speed
- Cut manual work and errors in test creation
- Adapt swiftly to various screen and UI designs

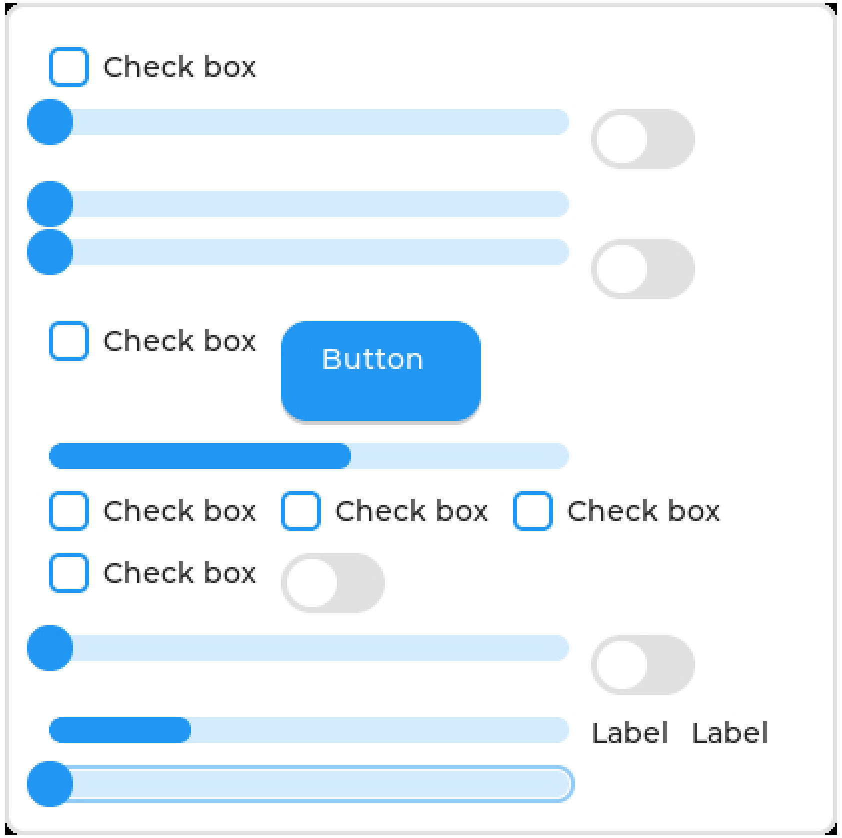
Research Questions

- How Does YOLOv8 enhance UI Widget Detection?
- What Challenges Does YOLOv8 Overcome in Automated Embedded UI Testing?
- How Effectively Does YOLOv8 Detect and Classify Widgets in Various UI Layouts?
- What Limitations of Current Testing Methods Does YOLOv8 Address?

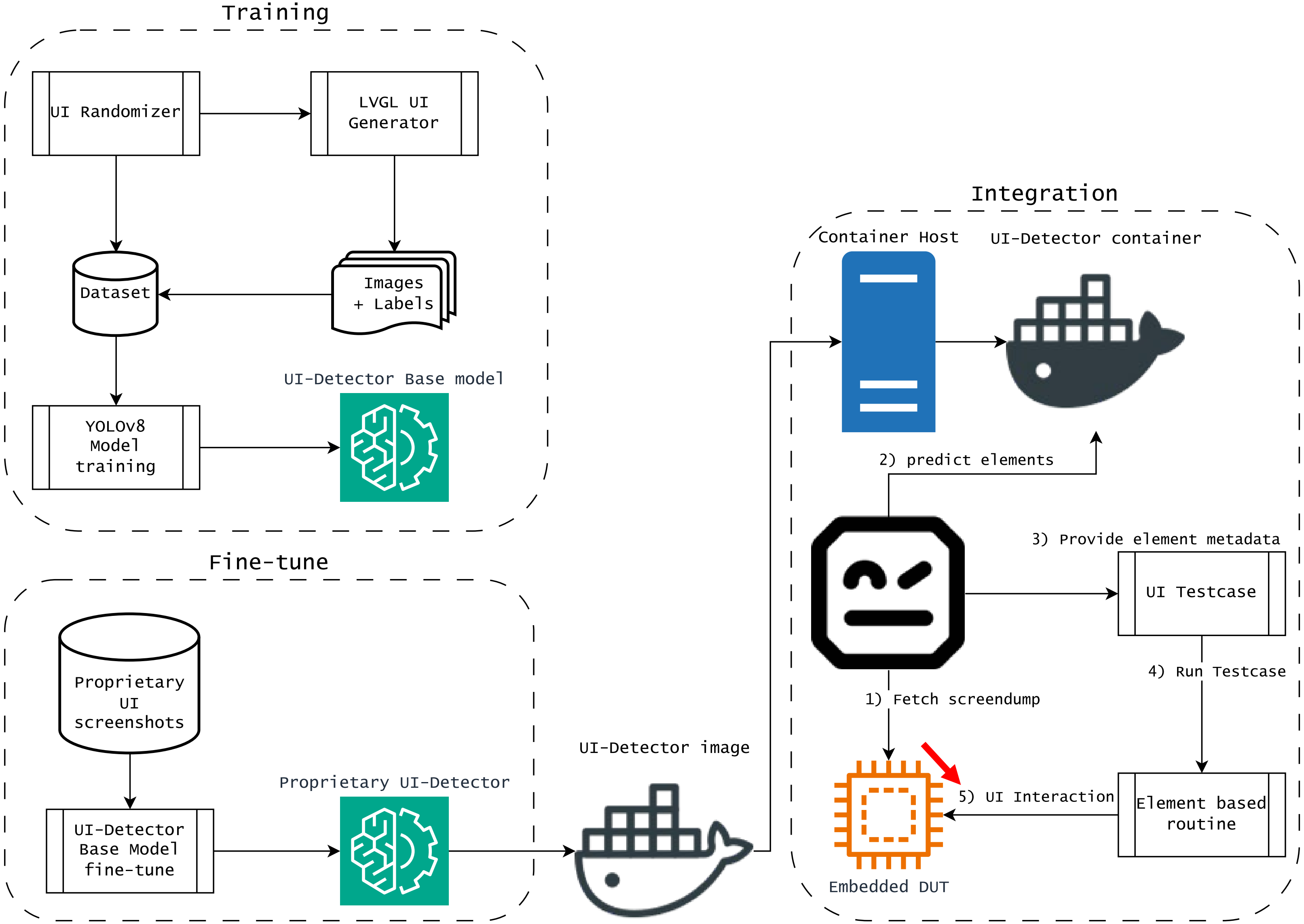
What is a widget?

- Basic building block of UI
- Elements on screen

  (buttons, text fields, checkboxes …)
- Testing Challenges
  - High variability in design
  - Responsive to screen size
  - Dynamic behavior

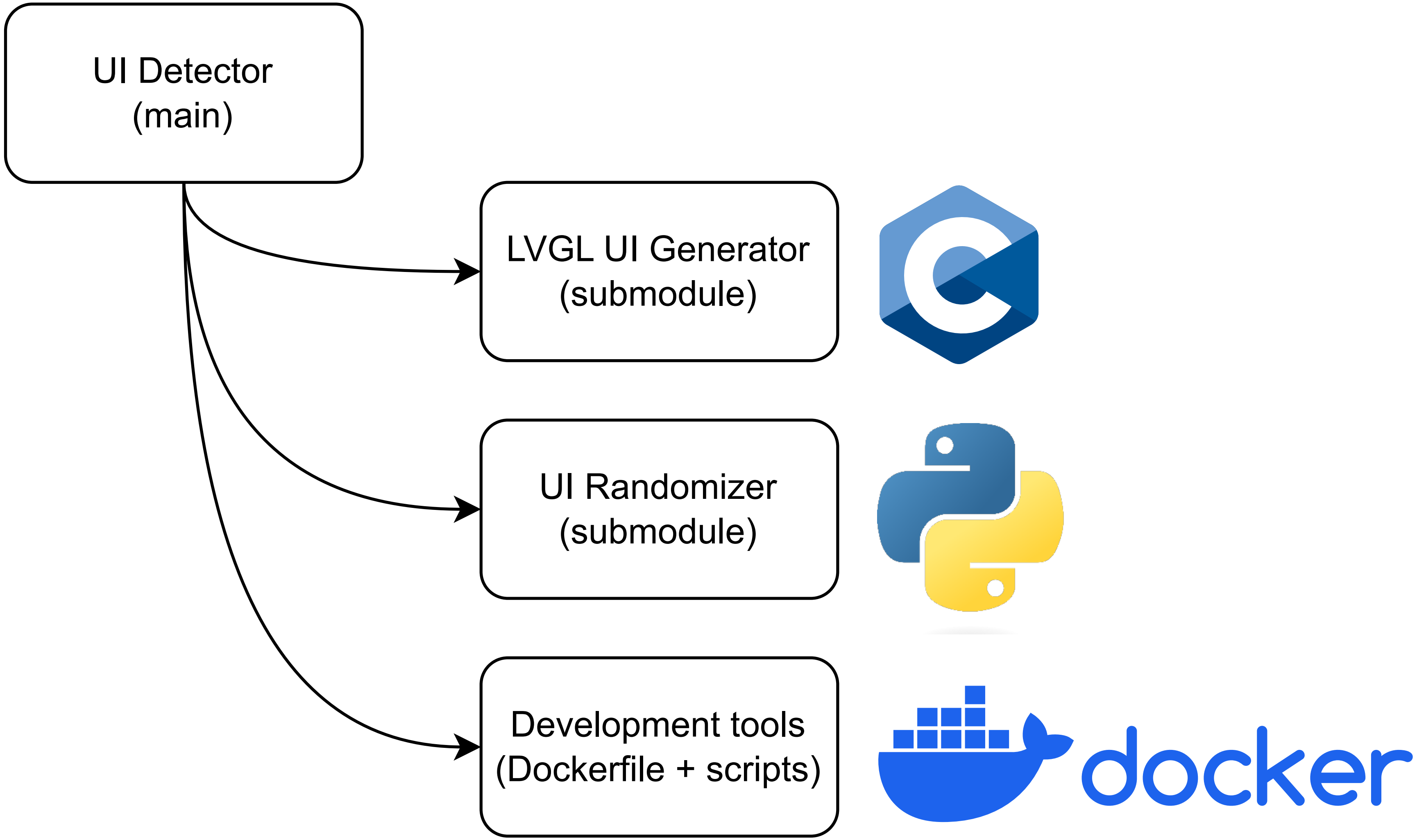


Proposed Solution



FH University of Applied Sciences TECHNIKUM WIEN

Project Architecture – Overview



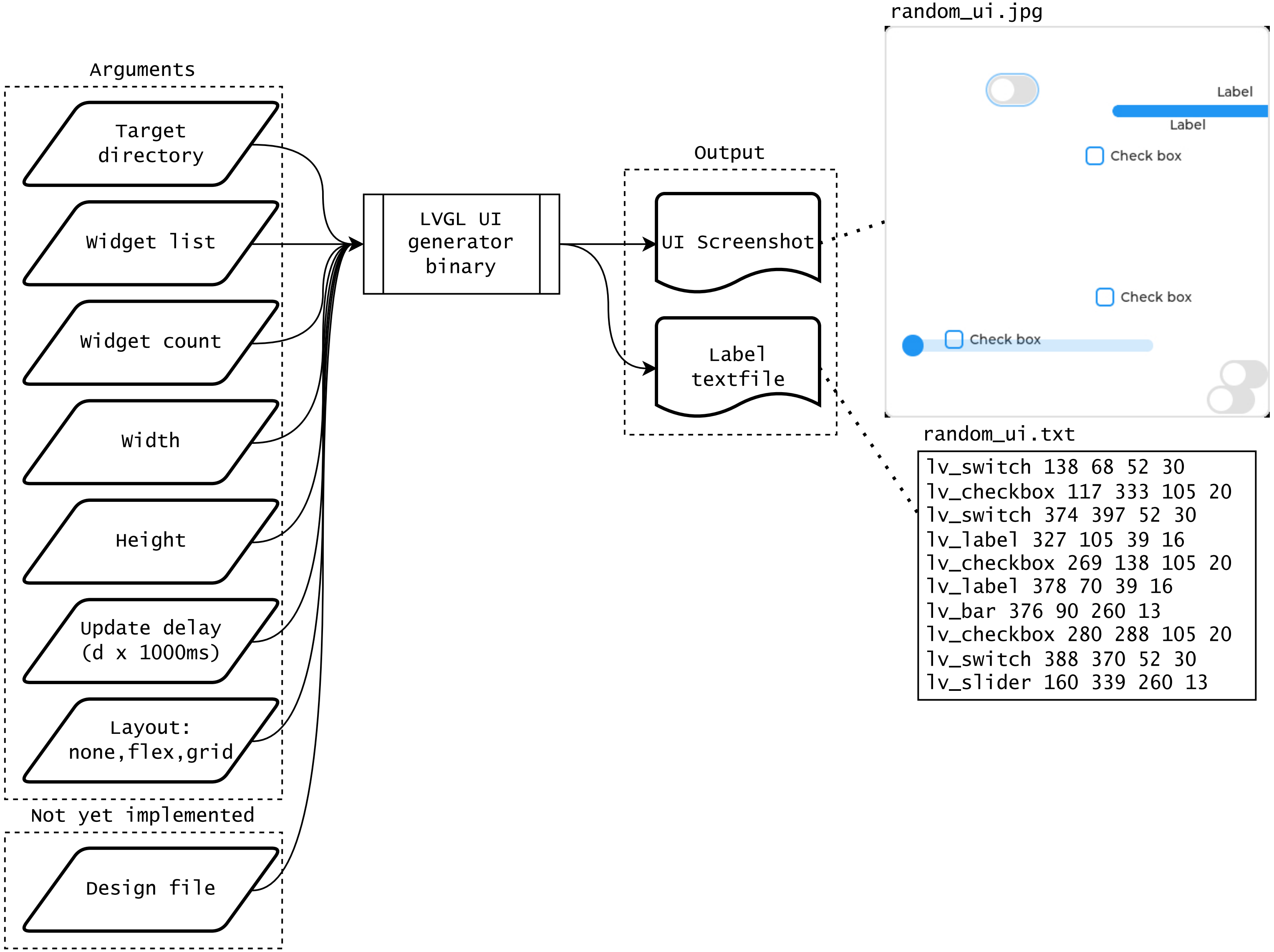
UI Detector (main)

LVGL UI Generator (submodule)

UI Randomizer (submodule)

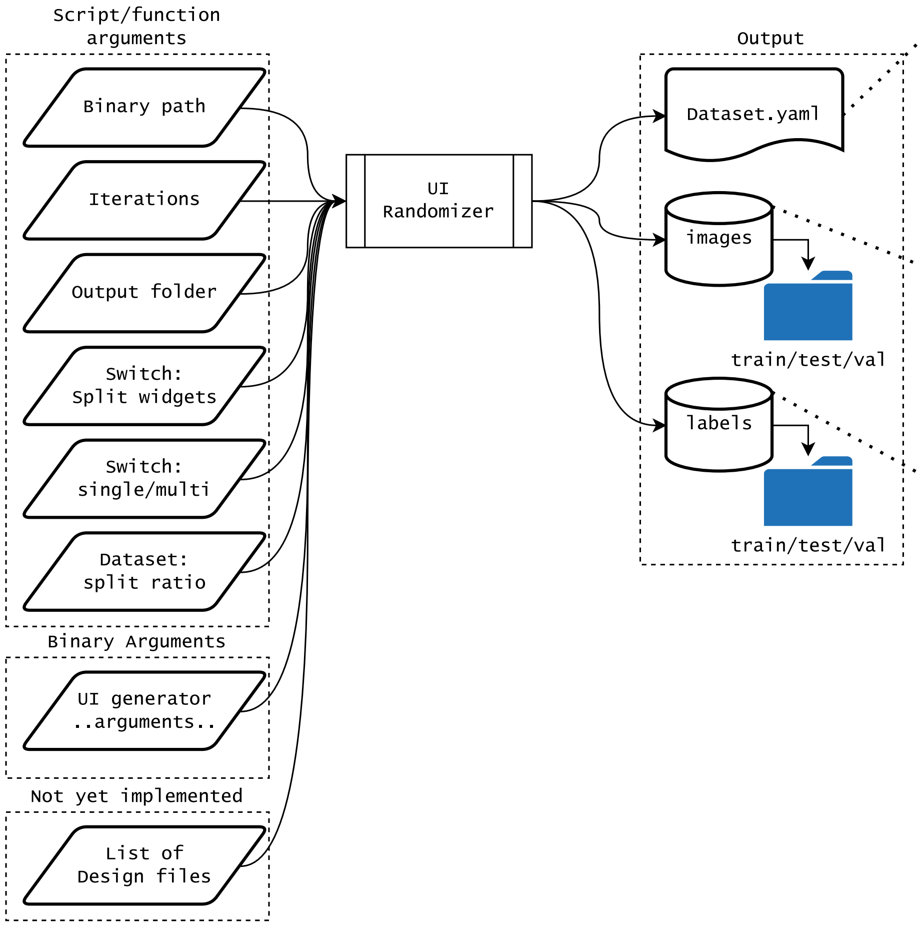
Development tools (Dockerfile + scripts)

Project Architecture – LVGL UI Generator



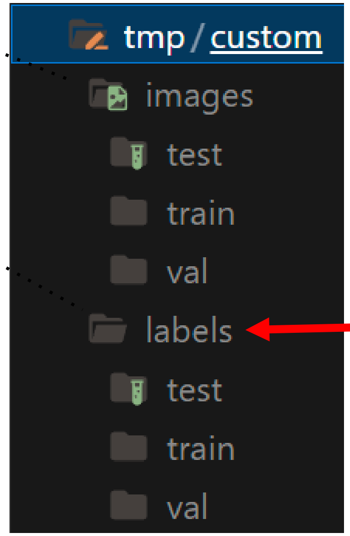
```
lv_switch 138 68 52 30
lv_checkbox 117 333 105 20
lv_switch 374 397 52 30
lv_label 327 105 39 16
lv_checkbox 269 138 105 20
lv_label 378 70 39 16
lv_bar 376 90 260 13
lv_checkbox 280 288 105 20
lv_switch 388 370 52 30
lv_slider 160 339 260 13
```

Project Architecture – UI Randomizer



random_ui.txt

```
names:
  0: lv_btn
  1: lv_checkbox
  2: lv_label
  3: lv_slider
  4: lv_switch
  5: lv_bar
path: /workspace/project/tmp/custom
test: images/test
train: images/train
val: images/val
```



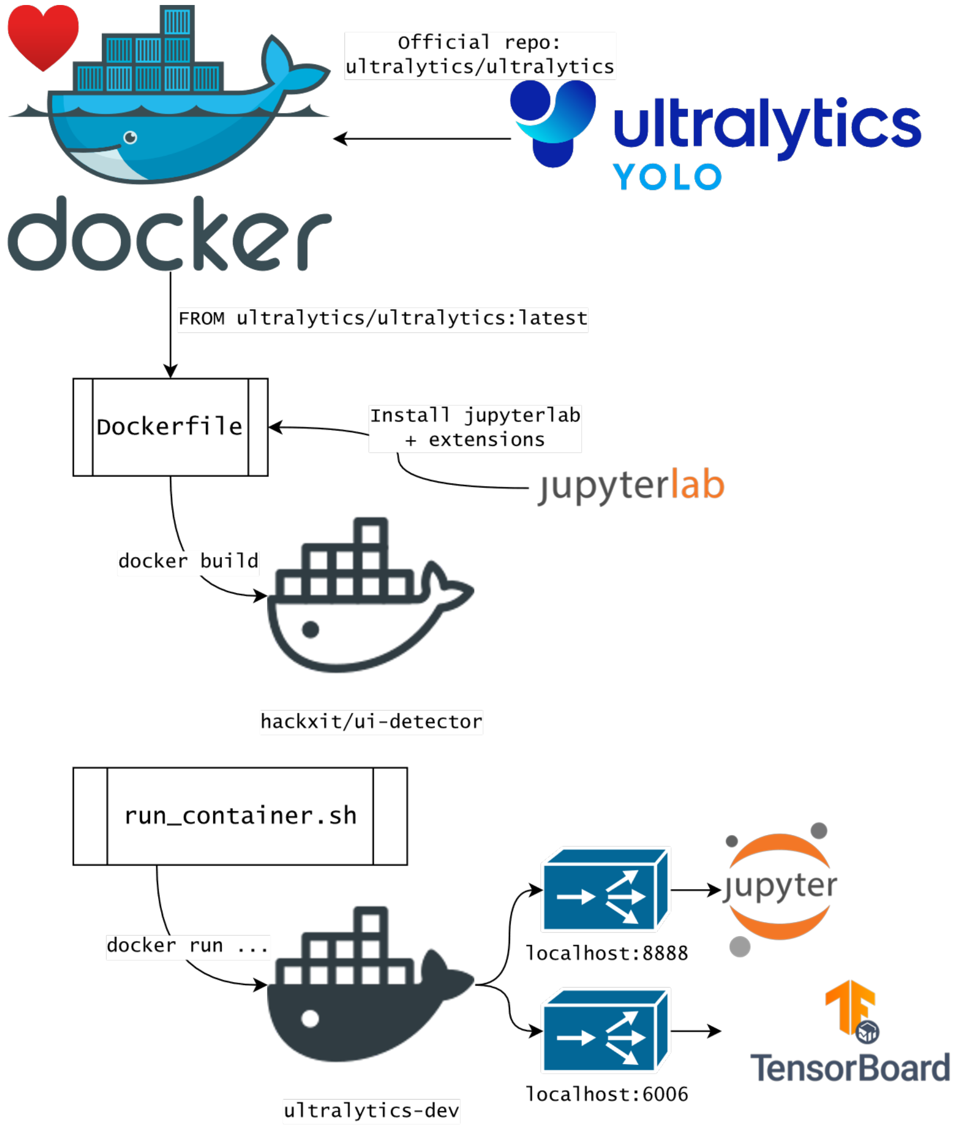
Pre-processing

Pre-processed random_ui.txt

```
4 0.3317307692307692 0.16346153846153846 0.125 0.07211538461538461
1 0.28125 0.8004807692307693 0.25240384615384615 0.04807692307692308
4 0.8990384615384616 0.9543269230769231 0.125 0.07211538461538461
2 0.7860576923076923 0.25240384615384615 0.09375 0.038461538461538464
1 0.6466346153846154 0.3317307692307692 0.25240384615384615 0.04807692307692308
2 0.9086538461538461 0.16826923076923078 0.09375 0.038461538461538464
5 0.9038461538461539 0.21634615384615385 0.625 0.03125
1 0.6730769230769231 0.6923076923076923 0.25240384615384615 0.04807692307692308
4 0.9326923076923077 0.8894230769230769 0.125 0.07211538461538461
3 0.38461538461538464 0.8149038461538461 0.625 0.03125
```

FH University of Applied Sciences TECHNIKUM WIEN

Project Architecture – Development container



Official repo: ultralytics/ultralytics

FROM ultralytics/ultralytics:latest

Install jupyterlab + extensions

docker build

hackxit/ui-detector

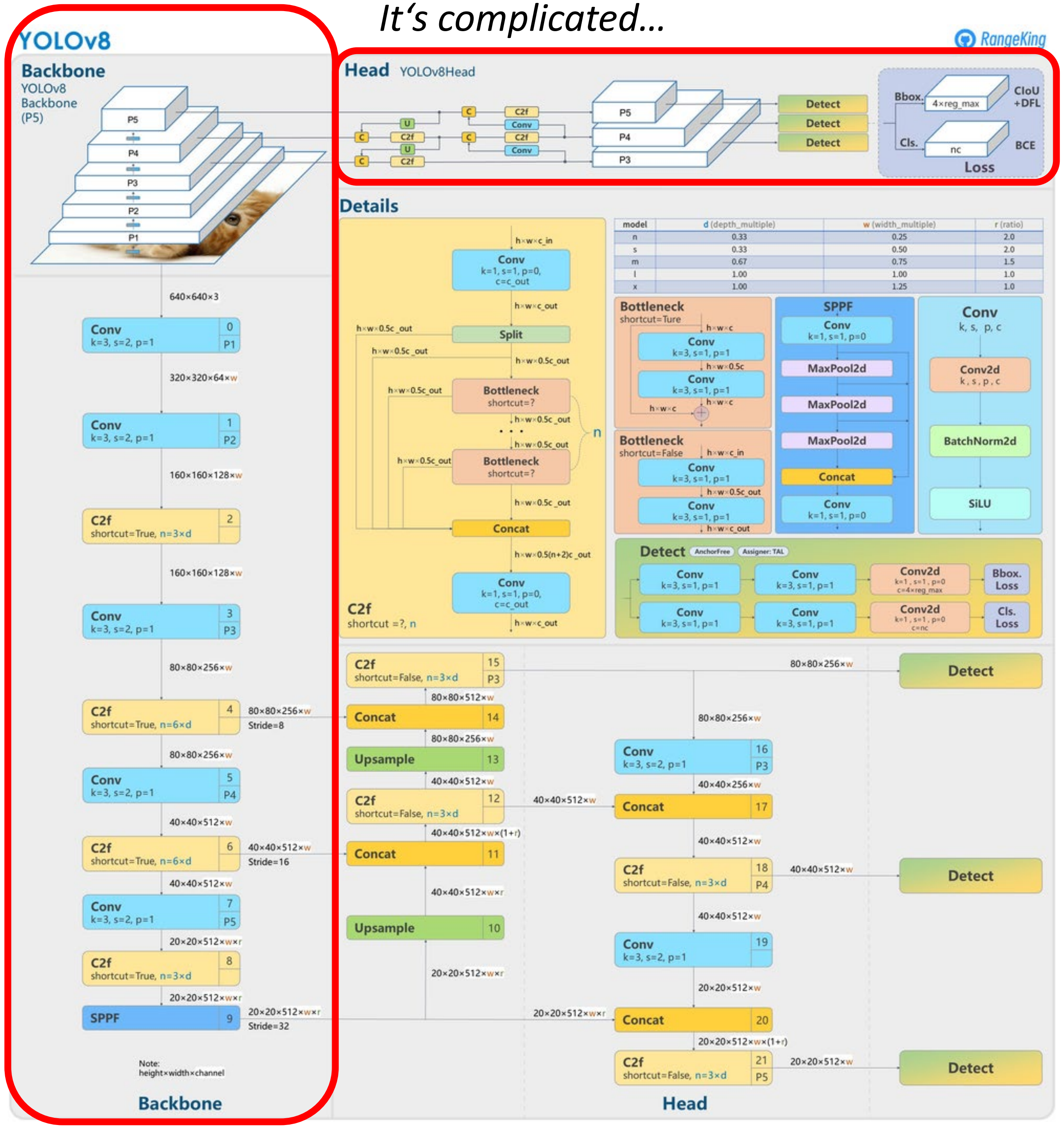
docker run ...

ultralytics-dev

YOLOv8 Architecture

- ***Y**ou **o**nly **l**ook **o**nce*
  - *One forward pass to predict every object*
- Backbone *(Feature Pyramid)*
- Head *(Detection/Classification)*

It's complicated…

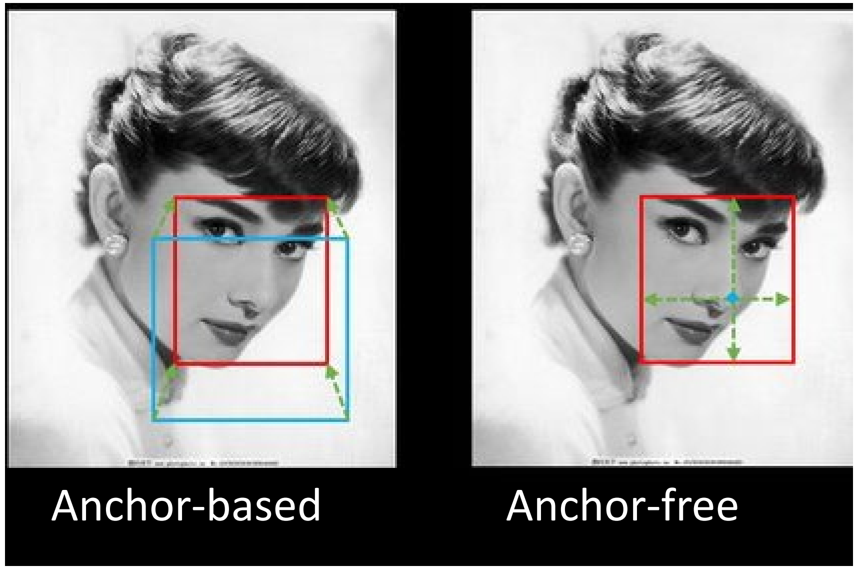


It‘s complicated…

YOLOv8 Architecture

- ***Y**ou **o**nly **l**ook **o**nce*
  - *One forward pass to predict every object*

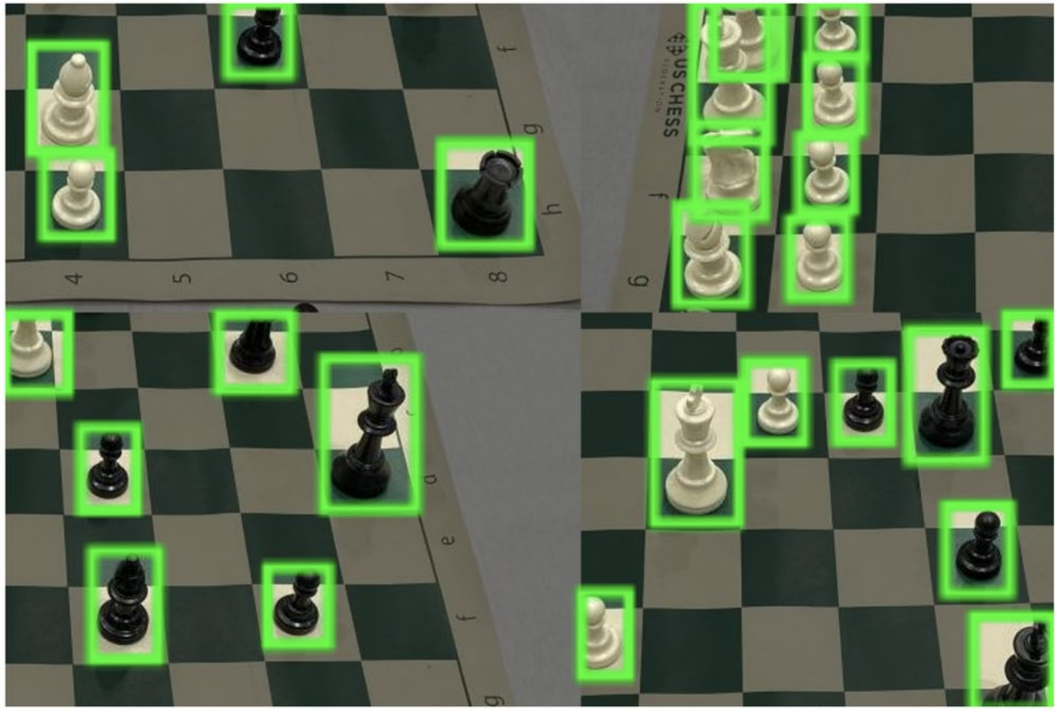
- Anchor-free detection



It‘s complicated…

YOLOv8 Architecture

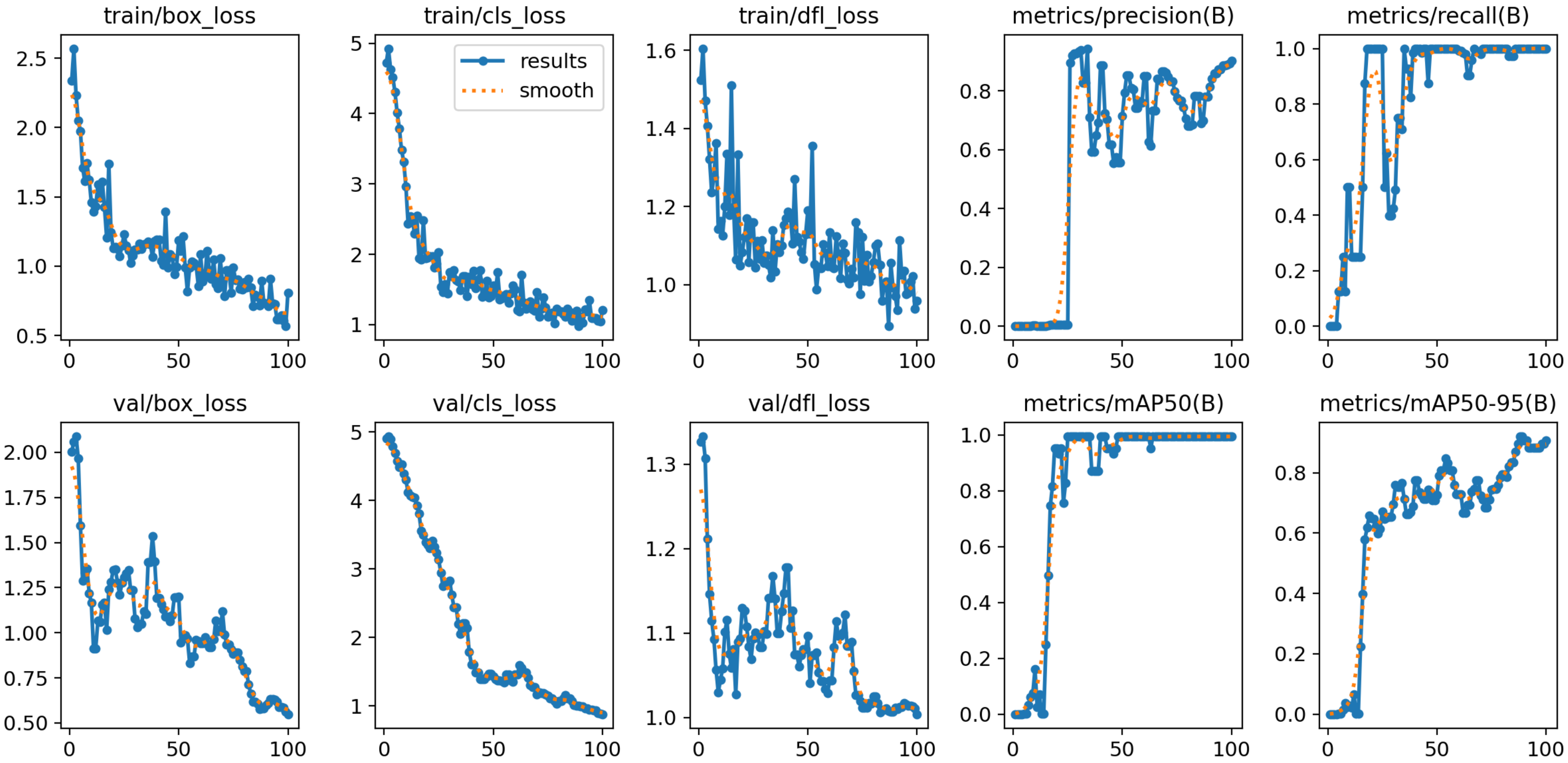
- ***Y**ou **o**nly **l**ook **o**nce*
  - *One forward pass to predict every object*
- Mosaic augmentation



Current project state & results

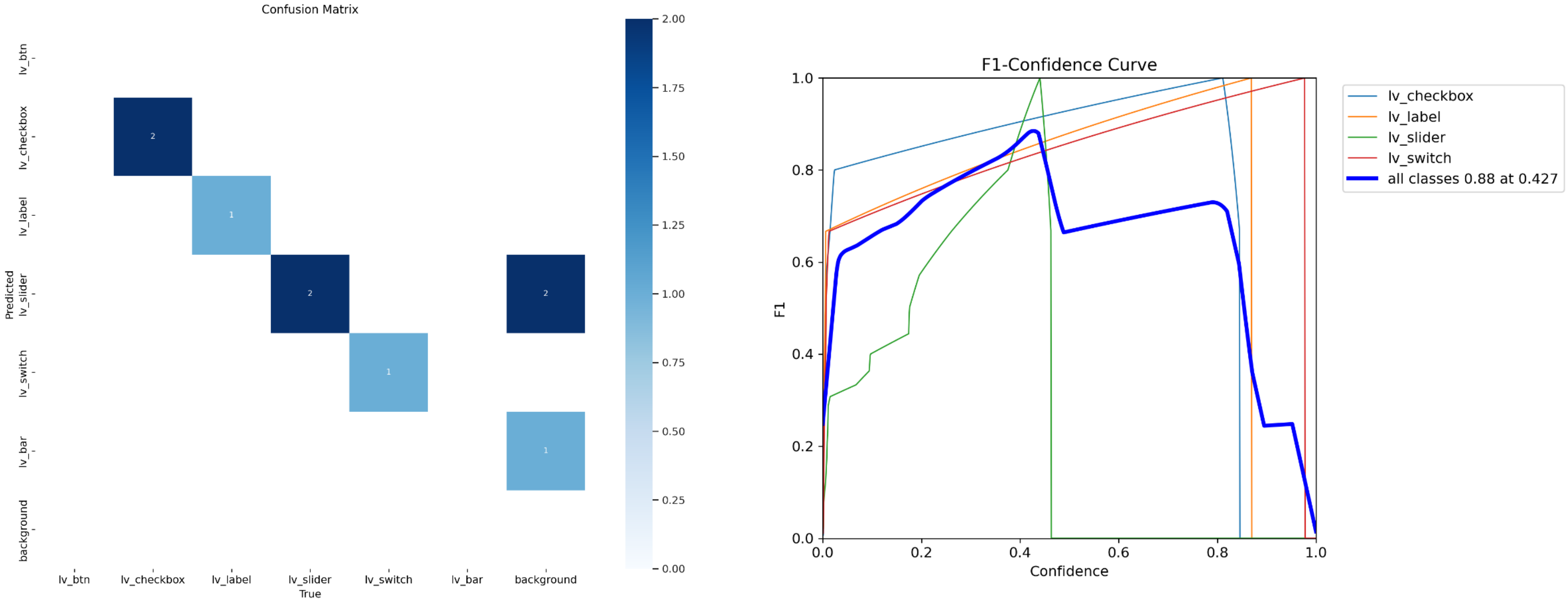
- https://github.com/HackXIt/lvgl-ui-detector/tree/3f7769dac85c5d8f3c70ad3b7c7c77cfc42d2a77
- Trained with 42 synthetic images
  (+12 test & +6 validation images)
- *Classes: button, checkbox, slider, switch, progressbar*
- Current result is not very realistic due to bad training data
- Synthetic dataset size will be larger once the generator gets updated

Current project state & results



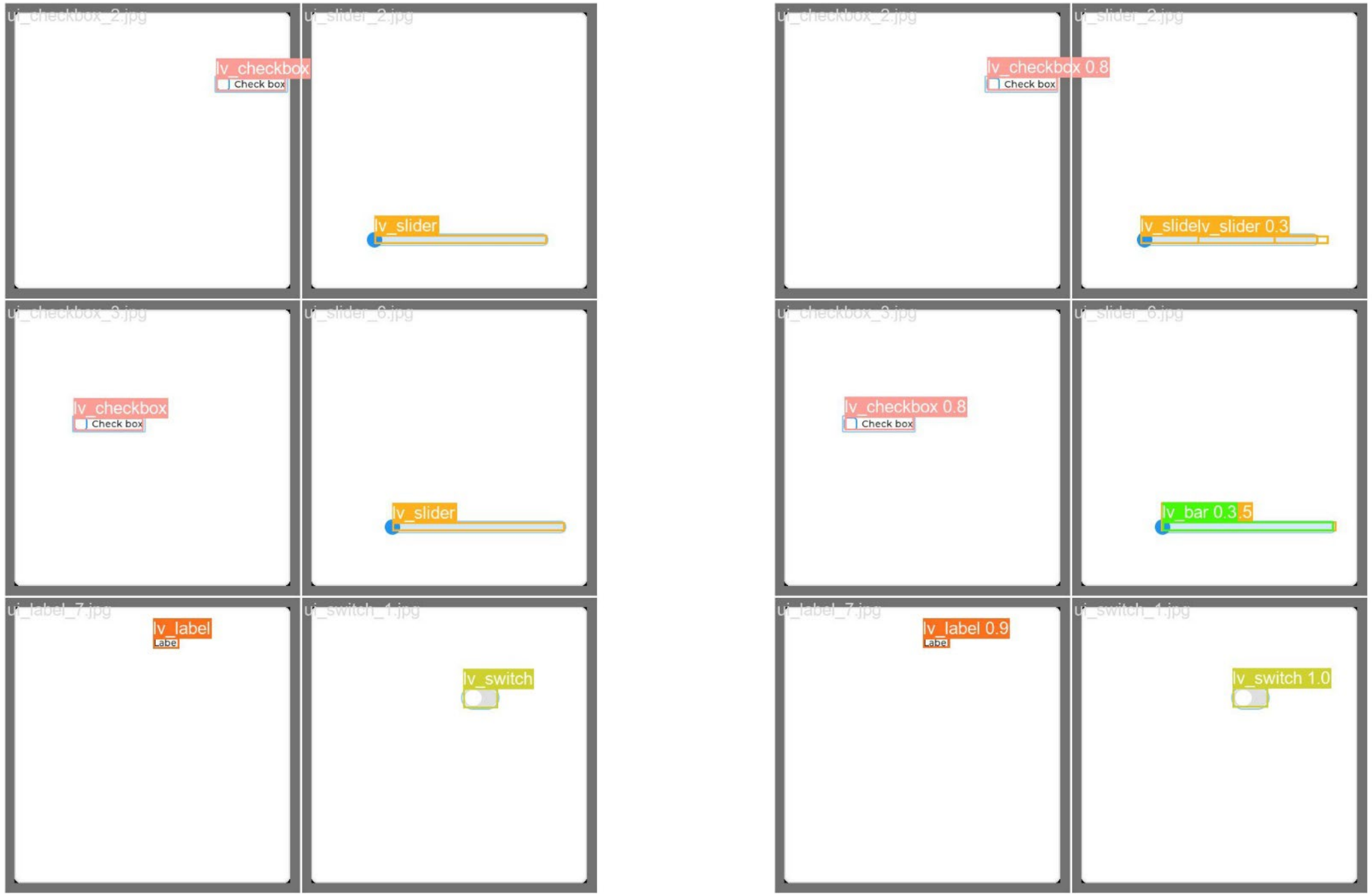
training results

Current project state & results



Validation results

Current project state & results



Validation results

Q&A

Image references

- [Slide 2] Schrack Seconet company logo provided by Nikolaus Rieder
- [Slide 4] Robot Framework test log provided by Nikolaus Rieder
- [Slide 4] Schrack Seconet product image from quick start guide
- [Slide 9-12] Project-Diagrams by Nikolaus Rieder
- [Slide 13] YOLOv8 Architecture by RangeKing
- [Slide 13] Image: Anchor-based vs. Anchor-free
- [Slide 13] YOLOv8 Mosaic Augmentation of chess board photos

Literature References

- [1] Ultralytics YOLOv8 repository & documentation
- [2] A comparison of YOLOv5 and YOLOv8 in the context of mobile UI detection
- [3] Object Detection for Graphical User Interface: Old Fashioned or Deep Learning or a Combination?
- [4] GUI Component Detection Using YOLO and Faster-RCNN
- [5] GUI Element Detection from Mobile UI Images Using YOLOv5
- [6] UIED: a hybrid tool for GUI element detection
- [7] You only look once: Unified, Real-Time Object Detection
- [8] Software Test Automation with Robot Framework
- [9] Usage of Robot Framework in Automation of Functional Test Regression
- [10] A comprehensive Study on Automation using Robot Framework

Editor‘s Reference – BACKUP SLIDES

These slides weren‘t part of the presentation

FH University of Applied Sciences TECHNIKUM WIEN

Model training status

- YOLOv8 is a valid choice according to initial training
- YOLOv8 training is much easier than anything else tried
- Synthetic data not very varied resulting in unrealistic results
- Still requires fine-tuning with proprietary UI *(data missing)*

Model integration

- The fine-tuned model is still a work in progress
- Integration in test procedure as an API
- Predictions will inform about proprietary elements and positions
- Test procedure will leverage known element type and positions to perform routine

Synthesizing data for the model

- LVGL simulator for PC
- Varied dataset generation for realistic user interfaces
- Implementation currently incomplete
  - Currently only performs random placement of widgets
- Work in progress
  - Randomized style properties for created widgets
  - Realistic placement based on design file
  - Include widget state variation