DOI: https://doi.org/10.36719/2663-4619/96/127-138

Shahriyar Guliyev
Nakhchivan State University
undergraduate
shahriyarguliyev@ndu.edu.az

RECOGNITION OF THE DISABLED, ELDER AND GAZIS IN DEEP CNN/YOLO NETWORK AND RESEARCH IN ORGANIZATION OF SOCIAL-PUBLIC SERVICE

Abstract

Many problems that exist today are of social importance. One of them is a special part of our society - members of the physically disabled and elderly class. The more we pay attention to them, the more we will rise as a society and the prosperity will increase. We can make the lives of those people easier by realizing that we will have to organize a separate service for these issues in the near future, where there will be no areas where technology will not infuse. We know that public and social areas are monitored through IP cameras already available in many places. Taking advantage of this opportunity, these special people can be presented special treatment in public transport, crossing, public areas, catering facilities, etc.

Keywords: *artificial intelligence, machine learning, artificial neural networks, deep learning, computer vision, object recognition, object detection, classification, localization, segmentation, Deep CNN, YOLO network, physically limited, elderly people, qazi detection*

Şəhriyar Quliyev
Naxçıvan Dövlət Universiteti
bakalavriat
shahriyarguliyev@ndu.edu.az

Əlil, yaşlı və qazilərin dərin CNN/YOLO şəbəkəsində tanınması və sosial-ictimai xidmətin təşkilində tədqiqi

Xülasə

Günümüzdə mövcud olan bir çox problem ictimai əhəmiyyət daşıyır. Onlardan biri də cəmiyyətimizin xüsusi parçası- fiziki məhdudiyyətli və yaşlı təbəqənin üzvləridir. Onlara hər nə qədər diqqət ayırsaq bir o qədər cəmiyyət olaraq yüksələrik, firavanlıq artar. Texnologiyanın demək olar ki nüfuz etmədiyi sahə qalmayacaq yaxın gələcəkdə bu məsələlərə də ayrıca xidmət təşkil etməli olduğumuzu dərk etməklə həmin insanların həyatını asanlaşdıra bilərik. İctimai-sosial məkanlarda bir çox yerdə mövcud olan IP kameralar vasitəsilə müşahidə aparıldığını bilirik. Bu imkandan faydalanaraq ictimai nəqliyyatda, keçidlərdə, sosial təsislərdə, iaşə obyektlərində bu özəl insanlara ayrıcalıq tanıyaraq, onlara xüsusi yanaşmaq olar.

Açar sözlər: *süni zəka, maşın öyrənməsi, süni neyral şəbəkələr, dərin öyrənmə, computer vizion, obrazların tanınması, obyektlərin aşkarlanması, siniflәndirmə, lokalizasiya, seqmentasiya, Dərin CNN, YOLO şəbəkəsi, fiziki məhdudiyyətli, yaşlı, qazilərin tanınması*

Introduction

Men die but their works endure.

This is the enhanced and the well-documented version of the project I've built during and for the Engineering competition on Heydar Aliyev 100th anniversary being held in French-Azerbaijan University in Baku in October 14-15, 2023. Helping the physically disabled, elderly people in transportation, helping them get a better life like us- the healthy people, is being considered. Apart from recent experiences in AI (Guliyev, 2023), now, for the first time in Azerbaijan- the preparation,

annotation, augmentation of a dataset of Gazis' (*veterans'*) images for the training in our ANN model in use, consisting of around 50 (augmented to 99) raw images, is carried out.

Proposal

• Providing the physically disabled people with special services, looking at the opportunities to apply them to public life.

• For example, the implementation of this system in the ASAN national service, or special switching mechanism and sounding activities in case of a disabled person detection at traffic lights, etc.

• Providing informative message in public transportation (like a bus or subway) to help them get a seat.

I. Contents & Implementation Plan

Data Feed Unit:

• Camera footage must be streamed over a communication channel (Wi-Fi, CAN bus, Ethernet layer, so on).

• For the test purposes a remote connected smartphone in the localhost can do.

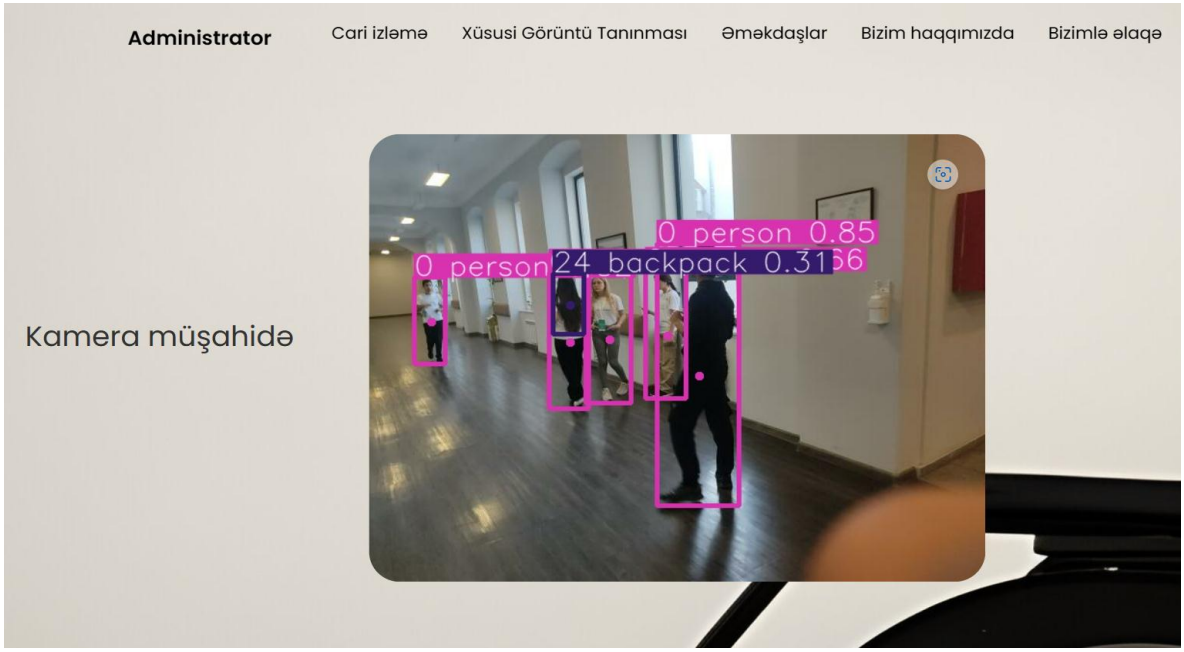

Figure 1. Generic YOLO model detection from Smartphone over DroidCam

Web Application & Inferencing:

• Web UI (responsive) for the administrative monitoring and management of the system.

• Another Web UI packaged into a Mobile Application (APK & IPA) for user/worker interaction on continuous data stream.

• The same server runs the inferencing job and adds bounding boxes, segmentation overlays on imagery data, etc.

Backend Server Units

1. Web UI (Admin)**:**

• I've built our app using VueJS (template-syntax) and Flask (backend) Python microframework.

• In the real system, backend is going to be PHP 8 due to the extensive and reliable feature set.

2. Mobile App (Worker)**:**

• Due to the fact that 2 days is less for native mobile application, packaged the Web UI using WebView and the Capacitor JS for both Android and iOS.

3. Image Inferencing:

• Currently the 640px image data is being processed in CPU (*no GPU in Intel laptop*), Core i7, 12 GB RAM, 1TB HDD.

• Takes max ~2000ms, min ~300ms.

Computer Vision:

Object Recognition

• Classification
• Object Detection
• Localization
• Instance Segmentation
• Regularization
• Normalization
• Object Tracking
• Object Pose Estimation

"Object detection is preferred over image classification since the latter only focuses on determination of object within the image without providing any locality information" (Hussain, 2023: 2).

Available Models

✓ YOLOv8 (Ultralytics)
• YOLO-NAS (fastest inference)
• EfficientDet (a variant of YOLO)
• DETR (MIT, Facebook AI)
• RT-DETR (Facebook AI)
• ViT (Vision Transformer)
• CNN
• R-CNN
• Fast R-CNN
• Faster R-CNN
• Mask R-CNN (an extension of Faster R-CNN)
• ViT & CNN hybrids
• etc.

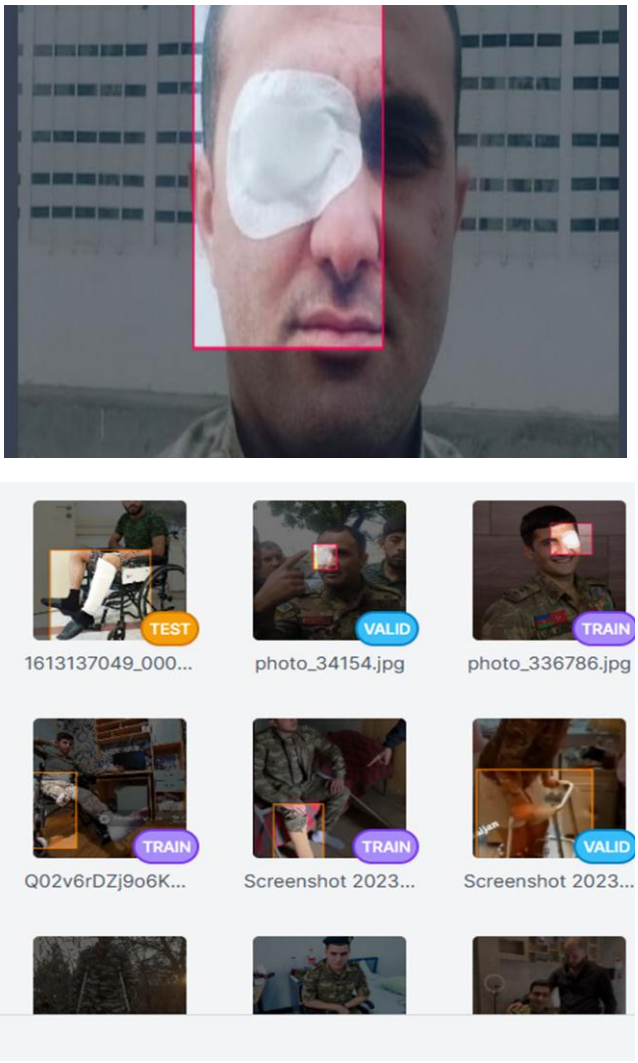

Figure 2. ***Object localization***

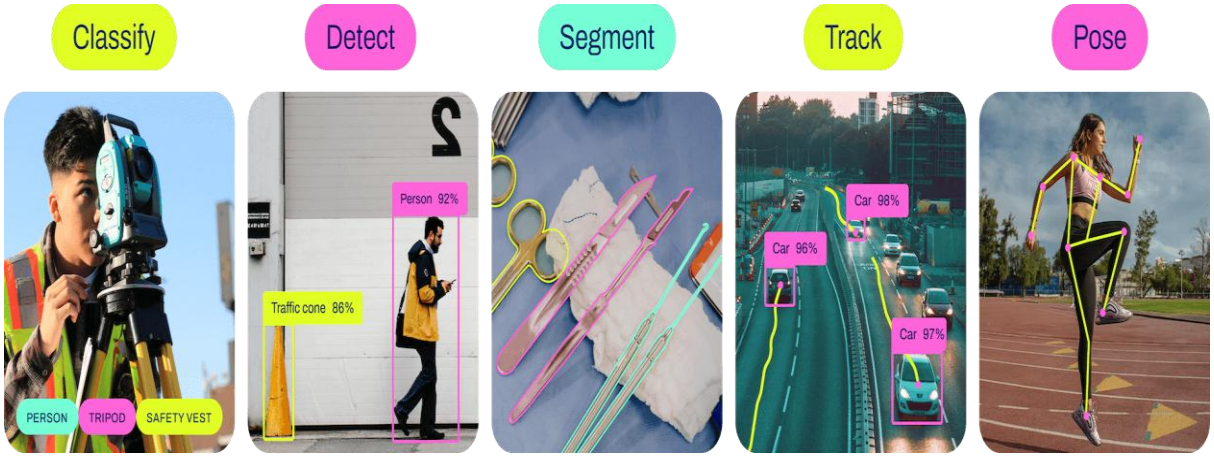

Figure 3. YOLOv8 features (11).

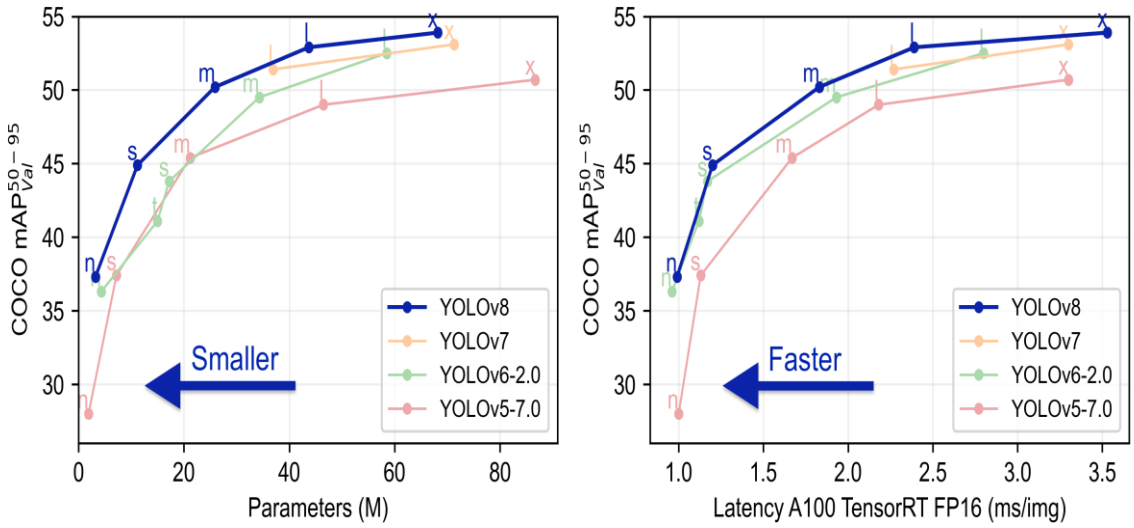

Figure 4. Performance of YOLO v8 in comparison with its predecessors (12)

I've chosen YOLOv8 CV ANN model because of the 48-hours of fast implementation, as it takes only 2.49 hours of specific-dataset (~2k size) training in the NVIDIA CUDA architecture (15GB mem) GPUs (T100, V100) and a slightly faster in the TPU of the Google Colab remote cloud computing processing units.

I. Dataset building & Training the model.

Before anything, must build a dataset:

Using Data Augmentation, I will increase the count of source images and prevent the overfitting issue of the ANN model. Basic methods of *Horizontal* and *Vertical flipping* of the images returns fine.

• Collecting source data for the dataset from open internet sources.

Tooling: **Selenium Automation Framework, BeautifulSoup** libraries.

**In contrast, a simple Roboflow Universe lookup brings excellent existing datasets shared by the open community, ready to use by just citing. I've also explored similar datasets for higher precision of the model.*

Datasets (abbrev. as DS)

1. Existent DS: BGVP (9)

This open-source dataset has 4 classes:

- Children without disability,
- Elderly without disability,
- Non vulnerable,
- With disability

Dataset Split:

- Train set: 1400 images (70% of overall images)
- Validation set: 399 images (20%)
- Test set: 196 images (10%)

Augmentations: No augmentations were applied.

Annotations: 5,914 (3.0 per image (average) across 4 classes).

Average Image Size: 0.25 megapixels (varying from 0.02 Mp to 24.00 Mp).

Median Image Ratio: 600x408 wide.

Class Balance:

- non_vulnerable: 2,518
- children_wo_disability: 1,644

Figure 5. BGVP sample

- with_disability: 939 (*under represented*)
- elderly_wo_disability: 813 (*under represented*)

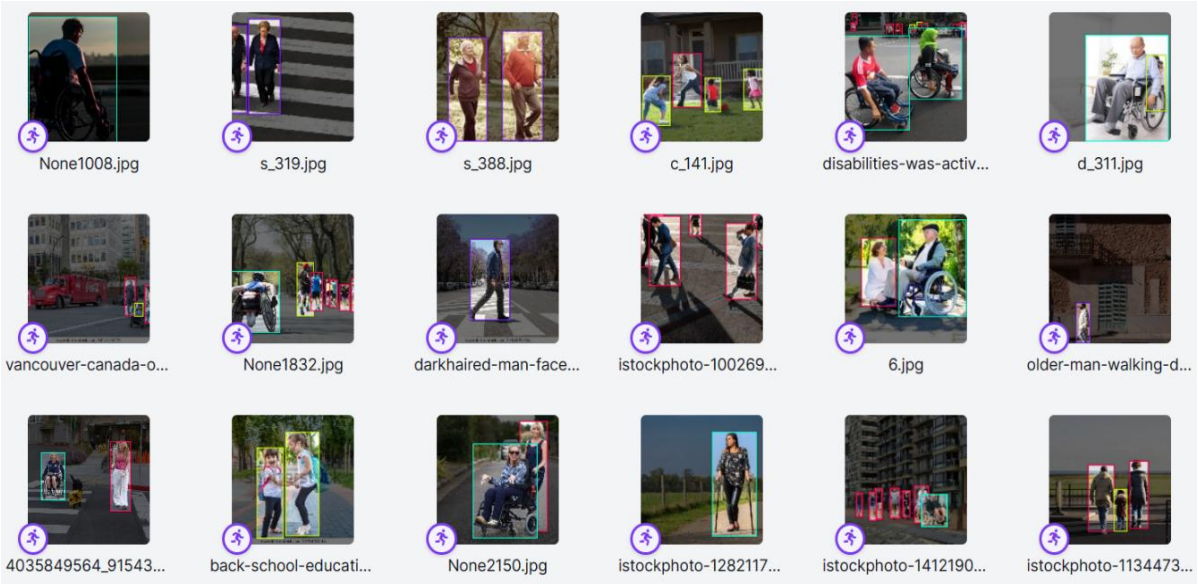

Figure 6. The BGVP dataset has 1995 total images.

Size distribution:

- small: 71
- medium: 1,420
- large: 382
- jumbo: 122 (> *1024x1024*)

Aspect Ratio Distribution:

- tall: 203
- square: 41
- wide: 1,746
- very wide: 5 (> *1.5:1*)

**The annotated data is available in the "TXT annotations" and "YAML config" used with YOLOv8.*

2. Novel DS: Qazi (10)

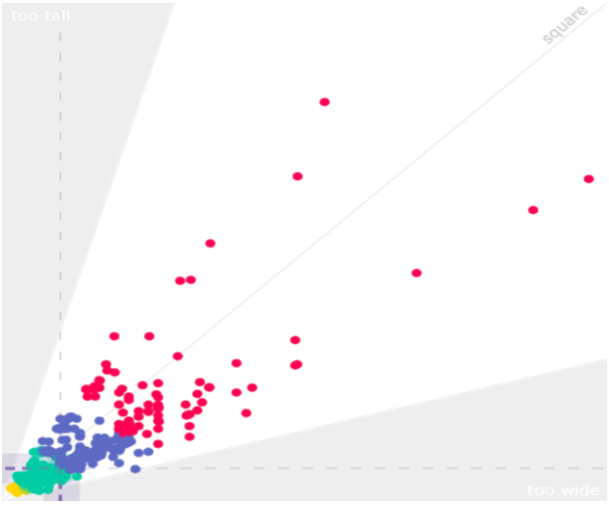

Figure 7. Size distribution (*purple box indicates the median width by median height img*)

Figure 8. Qazi recognition

To serve our heroes better, we have to recognize them. Initiating the idea, I've made a new dataset by images from open internet sources.

This dataset has these 4 classes:

- *ayağı-əlil* (leg disability),
- *gözü-əlil* (eye disability),
- *qolu-əlil* (hand/arm disability),
- *üzü-əlil* (facial/skeleton disability)

Dataset Split:

- Train set: 87 images (88% of all images)
- Validation set: 8 images (8%)
- Test set: 4 images (4%)

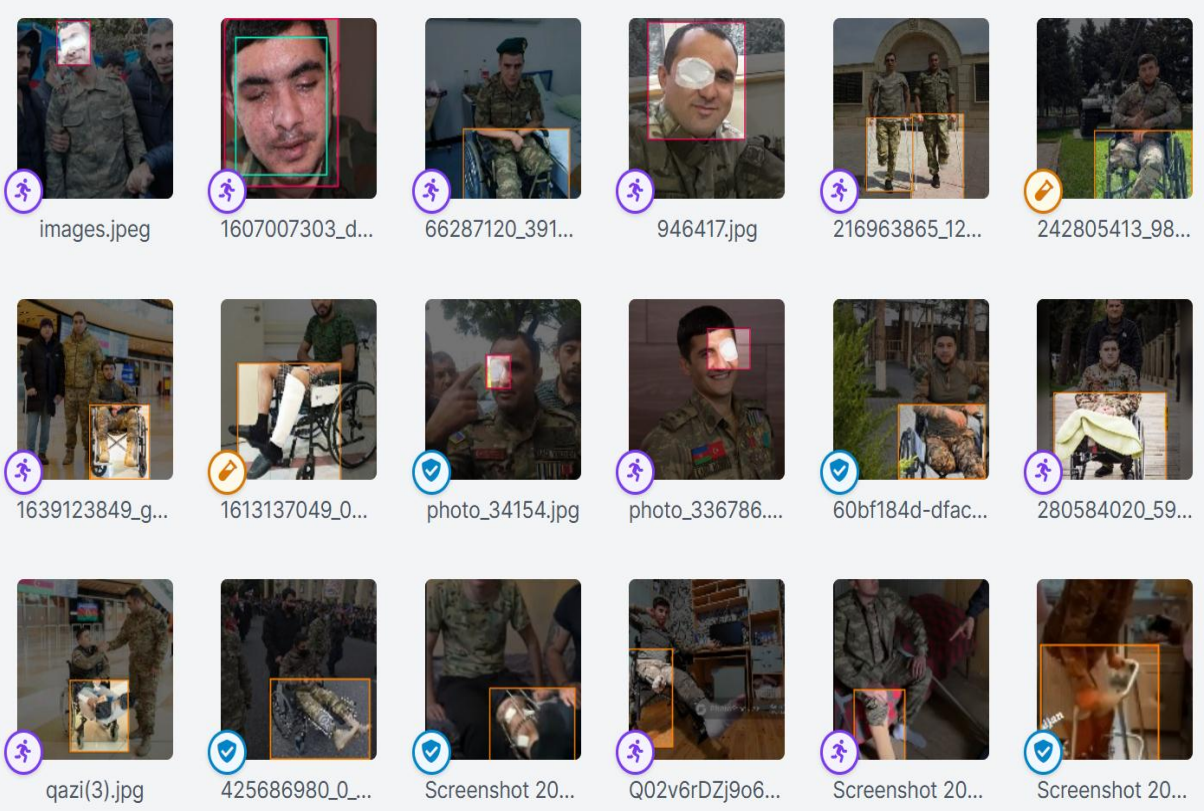

Figure 9. The Qazi dataset has total of 99 images

Preprocessing:

- Auto-Orient: Applied
- Resize: Stretch to 640x640

Augmentations:

- Outputs per training example: 3
- Flip: Horizontal, Vertical
- Crop: 0% Minimum Zoom, 20% Maximum Zoom

Annotations: 49 (1.2 per image (average) across 4 classes).

Average Image Size: 0.46 megapixels (varying from 0.05 Mp to 2.09 Mp).

Median Image Ratio: 720x600 wide.

Class Balance:

- ayağı-əlil: 32 (*over represented*)
- gözü-əlil: 12
- qolu-əlil: 4 (*under represented*)
- üzü-əlil: 1 (*under represented*)

Size distribution:

- medium: 10
- large: 24
- jumbo: 7 (> *1024x1024*)

Aspect Ratio Distribution:

- tall: 14
- square: 4
- wide: 23

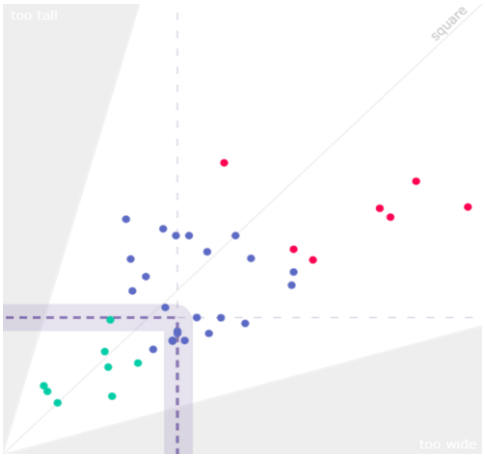

Figure 10. Size distribution (*purple box indicates the median width by median height image (600x408)*

Figure 11. A Qazi detection as eye-disability

The annotated data is available in the "TXT annotations" and "YAML config" used with YOLOv8.

Training the ANN model

Model summary: 225 layers, 11136761 parameters, 11136745 gradients.

- Computing Power: Tesla T4, 15102 MiB GPU
- Optimization algorithm: SGD (Stochastic Gradient Descent)
- Learning rate: 0.01

Training in the Cloud:

- Google Compute Engine backend (GPU)

Processing Unit:

As of our computer can't train in a day, have used Google Colab NVIDIA GPU.

- T4 15G VRAM processor (remote).

Initiating the training

I. Installing the requirements with dependencies:

@ultralytics==8.0.20 (from PyPI)

@roboflow==latest (from PyPI)

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.105.17   Driver Version: 525.105.17   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   65C    P8    11W /  70W |      0MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Figure 12. `nvidia-smi` command output outlining 15G of video memory

II. Importing YOLO and preparing the Dataset:

```
from ultralytics import YOLO
from roboflow import Roboflow
rf = Roboflow(api_key="API_KEY#")
project = rf.workspace ("myworkspace-IDENTITY_KEY#").project("DATASET_KEY#")
dataset = project. version (4). download ("yolov8")
```

III. Initiating the training in 350 iterations:

IV. @yolo COMMAND inline arguments

task=detect mode=train model=yolov8s.pt data={dataset.location} /data.yaml epochs=350 imgsz=800 plots=True

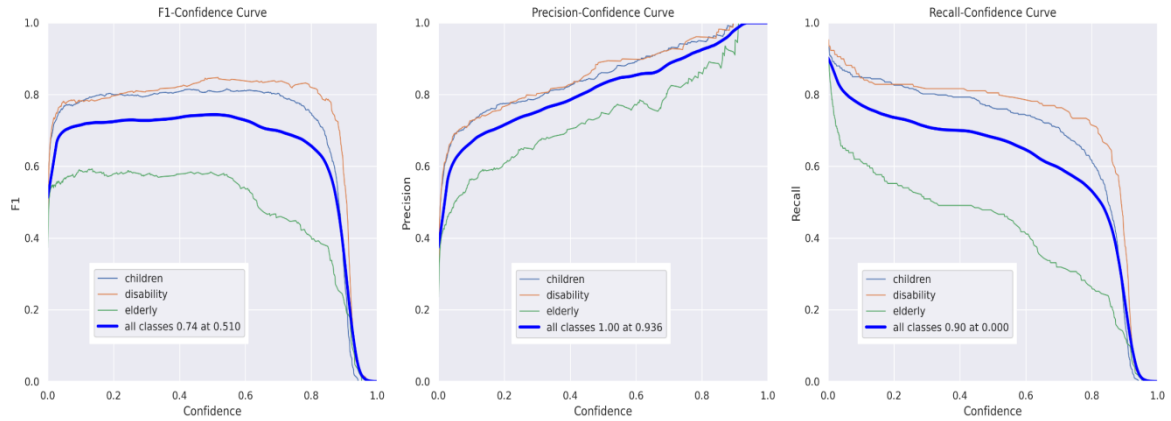

Figure 13. Confidence curvatures after the training

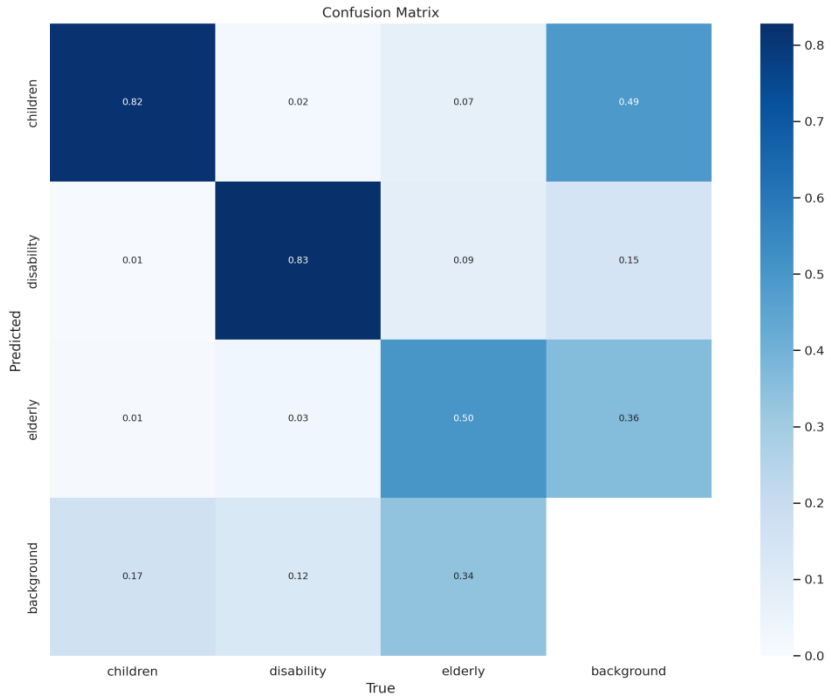

Figure 1413. Confusion Matrix after training in 350 iter

"For object detection task, the bounding box regression loss function plays a decisive role in localization performance" (Haiying, Cong, Yumei, Shuxiang, 2023: 7).

II. Feeding Imagery Data

Image is provided over remote server either from a camera or a video source.

Image Data Feeder:

- Mobile Smartphone camera in the localhost (*DroidCam in Android & EpocCam in iOS*),
- Laptop Webcam source (*optional*),
- Static/Live-stream Video upload from a source:
- For specific admin-analysis.
- External Webcam (*OpenCV compatible, such as Logitech QuickCam*).

In real systems, the imagery data source can be,

- CCTV video surveillance cameras,
- NVR security camera systems wireless (transmitting video to the recorder via Wi-Fi) and PoE (using Ethernet cable),
- or HVR in combination with DVR systems,
- etc.

III. Administrative UI Overview

Which runs the Inference and has control & statistical feature set.

Features:

- Live camera (source) detection (Figure 15).
- Specific (*static or live-stream*) Video (source) streamed recognition (Figure 16).
- Plotting daily and weekly object detection by each class in a graph.

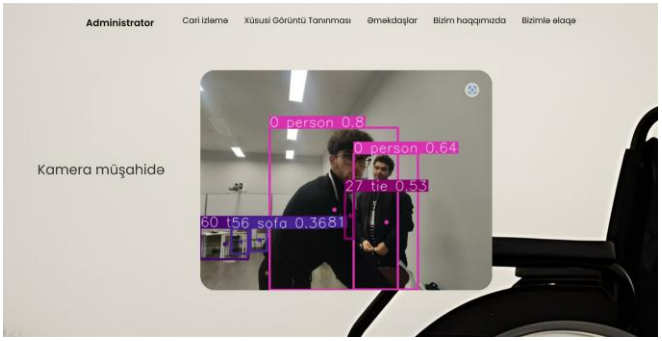

Figure 15. Realtime live camera (source) detection integrated to Admin UI

Data Storage

Figure 16. Object detection from a (static) video source

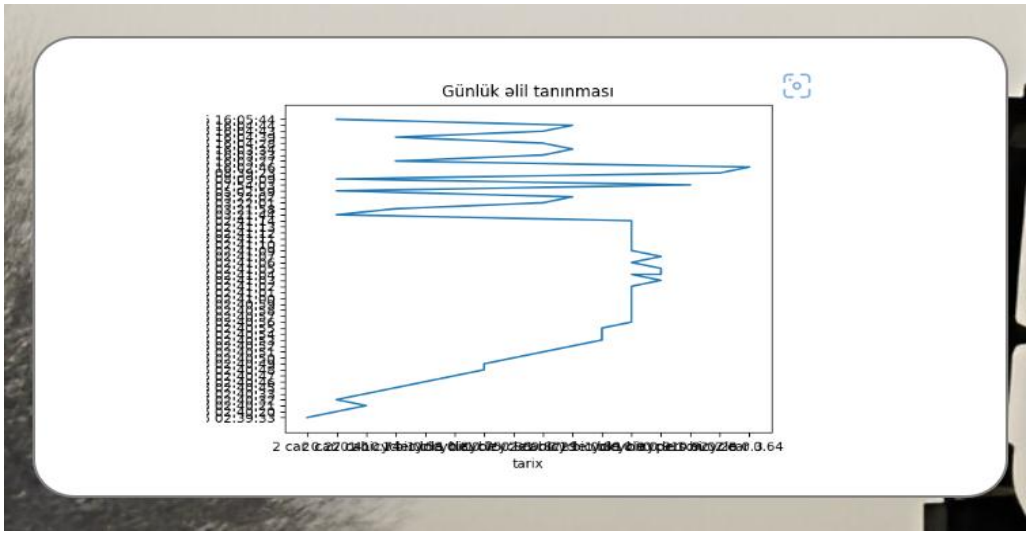

Figure 17. Realtime statistical plotting of detections in Admin UI
(stored to & fetched from the DB (SQLite used)

Database Management System:

• As it is a *pilot project*, using the mini, single-user SQLite is the option to go. But in a real system, ORDBMS like PostgreSQL or RDBMS like MySQL is the option to go because of the table-relations which return queries faster.

• To add enhanced management on data, object-relational ORDBMS system like PostgreSQL should be used.

Database & Table Structure:

• I have a single Database named "*ElillereXidmetMelumatlari*".

And 3 Tables are required:

• `user-accounts`: holds user and privilege information

	sira	novu	tarix ▲1
	Filter	Filter	Filter
1	612	1 bicycle 0.53	1697378625.46494
2	611	0 person 0.31	1697378620.25328
3	610	1 bicycle 0.37	1697378553.44708
4	609	2 car 0.4	1697378506.29422
5	608	0 person 0.27	1697378500.4195
6	607	2 car 0.26	1697378491.48131
7	606	1 bicycle 0.86	1697378476.81048
8	605	0 person 0.71	1697378474.81304
9	604	2 car 0.3	1697378473.53029
10	603	1 bicycle 0.61	1697378472.14878
11	602	2 car 0.4	1697378470.77102
12	601	1 bicycle 0.86	1697378458.44344
13	600	2 car 0.3	1697378456.33016
14	599	1 bicycle 0.86	1697378413.53387
15	598	2 car 0.48	1697378410.60889

Figure 1814. Sample data in the DB (*UNIX epoch time set*)

• `*(used) detection-information`: holds basic detection data entries (Figure 18)
• `detection-loc-history`: holds detected locations

Inference, Pre-/Post-Processing

• *Inference rates*: Up to 300 ms delay in Intel Core i7 CPU @12G RAM to infer a single frame of a stream.

• *Pre-Processing*: I compress images into 640px sizes while building a dataset for the relatively faster training in short period of time.

• But in the real system, I will not do much Pre-Processing to the image data byte-by-byte. It can infer at higher resolutions like 1080px (or higher).

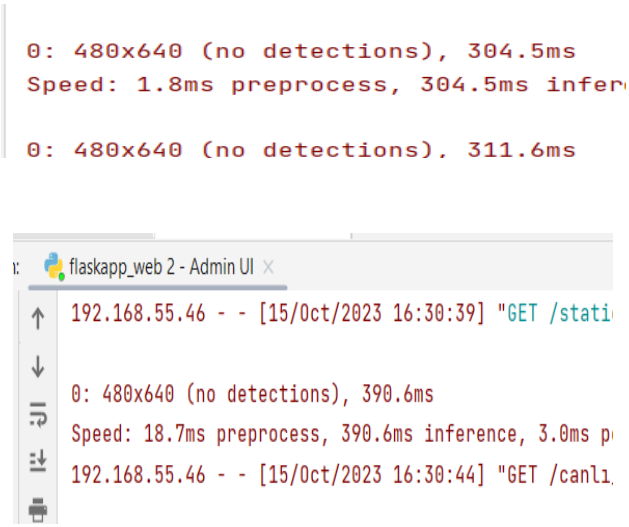

Figure 19. Inference speed is ~ 300 milliseconds in the Intel CPU

Post-Processing Algorithm:

The Inference Unit programmed in the Python ASL/PL. The procedure is as:

1. *Read* Imagery Data from a live Camera source (*VideoCapture* by ID supported by OpenCV library) or from a static, streamed video reels.

2. *Fetch* it frame-by-frame iteratively.

3. *Process* the image in the frame using the YOLO network's ANN model in efficient Weights state (*having higher precision score*).

4. *Add* bounding boxes, labels and draw onto the image frame using the OpenCV methods and yield the img.

5. *Encode* as JPEG and convert to bytes.

6. *Yield* the continuous frames (*rather than returning one-by-one*).

7. *Post* the image data as in Response (as Content-Type: image/jpeg) to the recipients (*img* element's *src* attribute *url_for* entry) to be shown in the Frontend UI as post-processed with overlays and added metadata.

Note: Instead of on-screen displays, can also export it as a video file (or both).

• The used OpenCV methods are *rectangle*, *getTextSize*, *putText* for overlaying the source image with detection information; And *VideoCapture*, *VideoWriter*, *VideoWriter_fourcc* methods for input/output manipulations.

IV. "Əlil Dostu" & "Əlillərə Xidmət" UI Overview

In this interface, I propose to attract volunteers from community who would like to help disabled and elders (earns cashback).

Mobility, Simplicity, Responsiveness:

• I also have a mobile application both for "*Əlil Dostu*" and "*Əlillərə Xidmət*" platforms.

IV. The Proposed Information System's Security Management

SAST (Static Application Security Testing):

• SonarQube in-cloud SaaS (SonarCloud) (8) for the GitHub VCS system integration before deployment.

• Code smells by Snyk (9) for dependency analysis.

• Ensure OWASP Top 10 & SANS/CWE Top 25 (10) compliant program code.

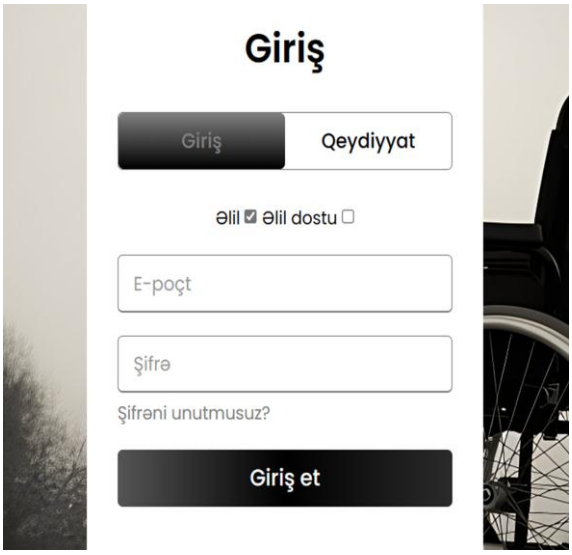

Figure 2015. Mobile UI has 2 login

It can be provided in case of it might rise a security concern. As if any program code or an environmental script or a database query might be vulnerable to those security standard, it might fail the QA (Quality Analysis) gate in the VCS/CI-CD Continuous Integration and Continuous Deployment job which might keep the overall production system secure.

DAST (Dynamic Application Security Testing):

• Proposed in the future.

Future Research Directions.

It can be enriched and enhanced to stream many more facilities benefiting more efficient inference algorithms and much better optimized ANN models. For now, looking for the implementation of such Information System in a pilot organization or governmental body as it is a social-public service offering.

I would like to explore more of Object Recognition tasks in different realtime effective architectures like Mask R-CNN as it has been also extensively used in recent years. "Mask R-CNN is yet another model used in image detection and segmentation tasks, which extends the Faster R-CNN architecture. If Faster R-CNN has only two outputs, the bounding boxes and the corresponding classes, Mask R-CNN also provides, in parallel, the segmentation masks" (Soviany P. and Ionescu R. T., 2018: 210/2).Also, YOLO-NAS is also performant from the benchmarks I've seen, that's why would also try it. And RT-DETR- after has been stabilized in the near future; would worth to try. And else than object detection, it would be more of challenging to explore facial recognition algorithms and optimization methods on this active topic.

Acknowledgements:

I would like to specially thank to the Engineering competition team members *Huseyn Oktyabrski* for building the frontend web page skeletons, *Onur Bayramov* for downloading the Qazi dataset images from open internet sources and *Mehemmed Haciyev* for annotating the squared labels on the Qazi dataset images. And special thanks to *Mohammad Moin Faisal* from whom have learnt very much knowledge on the YOLO network.

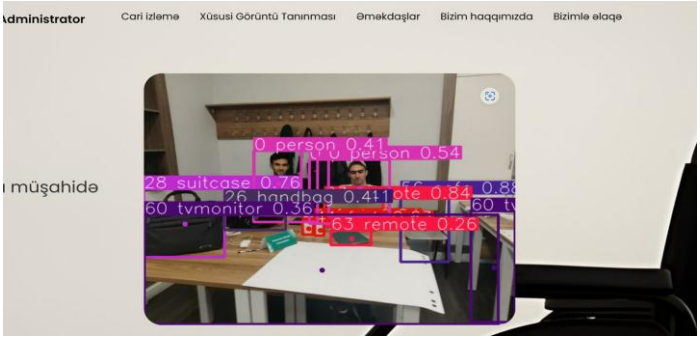

Figure 16. Realtime object detection with multiple

Conclusion

Figure 22. Əlil-card QR scanner

- I have prepared 3 AI-enabled UI – "**Admin**" (web-based), "**Əlil-dostu**" mobile application for the people *volunteering to help the disabled* and the "**Əlillərə-xidmət**" mobi le app for the disabled to *reach/access the available services*.
- With proposal of QR code for cards to, have used OpenCV again for URL dispatching (Figure 22).
- For **Əlil-dostu** UI have prepared Map (Figure 23) Geolocationing (used Google Maps).
- I want to bring more service to the disabled in the future.

Figure 23. Əlil-dostu GPS access

References

1. Hussain, M. (2023). YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. Machines, 11, 677 p. https://doi.org/10.3390/machines11070677. 25 p.
2. Soviany, P., Ionescu, R.T. (2018). Optimizing the Trade-Off between Single-Stage and Two-Stage Deep Object Detectors using Image Difficulty Prediction. 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, Romania, pp.209-214, doi: 10.1109/SYNASC.2018. 00041.
3. Guliyev, S. (2023). Artificial text generation using Deep Neural Networks: training of Suleyman Sani Akhundov's plays. Scientific Work: Volume: 17 Issue 6, pp.82-96, doi: 10.36719/2663-4619/91/82-96.
4. My Workspace. (2023). BGVP Dataset. Open Source Dataset. In: Roboflow Universe. Available at: https://universe.roboflow.com/myworkspace-lf7cf/bgvp.
5. Guliyev, S. (October 2023). Qazi Tanima. Open Source Dataset. In: Roboflow Universe. Available at: https://universe.roboflow.com/shahriyar-guliyev/qazi-tanima.
6. Haiying, X., Cong, Y., Yumei, T., Shuxiang, S. (2023). A dataset for the visually impaired walk on the road. Displays, Volume 79, 102486. ISSN 0141-9382. https://doi.org/10.1016/j.displa.2023.102486.
7. https://www.analyticsinsight.net/unveiling-top-10-databases-for-ml-and-ai/.
8. https://docs.sonarsource.com/sonarqube/latest/.
9. https://docs.snyk.io/snyk-api/snyk-rest-api-overview.
10. https://owasp.org/www-project-top-ten/,https://www.sans.org/top25-software-errors/, https://cwe.mitre.org/top25/.
11. https://raw.githubusercontent.com/ultralytics/assets/main/im/banner-tasks.png.
12. https://raw.githubusercontent.com/ultralytics/assets/main/yolov8/yolo-comparison-plots.png.

Received: 12.08.2023 Accepted: 04.11.2023