IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 7 August 2023, accepted 19 August 2023, date of publication 28 August 2023, date of current version 1 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3308600

RESEARCH ARTICLE

Evaluation of Human Pose Recognition and Object Detection Technologies and Architecture for Situation-Aware Robotics Applications in Edge Computing Environment

PEKKA PÄÄKKÖNEN AND DANIEL PAKKALA
VTT Technical Research Centre of Finland, 90571 Oulu, Finland

Corresponding author: Pekka Pääkkönen (pekka.paakkonen@vtt.fi)

This work was partially funded by Business Finland as part of the Next Generation Mining (NGMining) Co-Innovation Project.

ABSTRACT Realization of situation-awareness for autonomous robotics applications in edge computing environment is challenging. First, computing capabilities of edge devices are limited, which must be considered in the execution of machine learning (ML)-based solutions. Second, many technologies are available for realizing situation-aware capabilities, but comparison and integration of solutions creates additional challenges. Third, existing ML-based models are often not directly applicable for realizing custom applications, and model(s) may need to be re-trained with new data. The contribution of this paper is efficiency and feasibility evaluation of human pose recognition and object detection technologies in edge computing environment. Several lessons learnt covering constraints are presented regarding feasibility of the experimented technologies and data sets. The efficiency evaluation results indicated that simultaneous human pose recognition (Google's Movenet) and object detection (Yolov5) on Jetson AGX Xavier achieved ∼13-16 FPS, while GPU and CPU utilization remained at a medium level, and most of the memory remained unused (< 44 %). Object concept and human pose concept activation algorithms may be considered as an additional contribution. Realized architecture design of the prototyped system in multiple computing environments can be considered as a partial evaluation of a ML-based big data reference architecture.

INDEX TERMS Movenet, Yolov5, Jetson AGX Xavier, inference, reference architecture, big data.

I. INTRODUCTION

For effective cooperation of people and machines in future physical work, remotely operated autonomous robotic systems (e.g., automated guided vehicles (AGV)) should be able to safely operate in the same physical space with human employees without safety zones separating the two (e.g., collaborative robotics [1]). To achieve this, the robotic systems should have capabilities to perceive and comprehend their surrounding operational context and take it into account in the automated decision making as part of the missions or tasks execution. In other words, the future autonomic robotic systems need to become situation-aware just like humans, capable to perceive critical factors from their environment, understand what those mean in relation to current goals [2]. Accordingly, simple object detection is not sufficient, but perception of concepts from the environment is required to enable cognitive robotic systems [3]. For example, let us consider a use case of underground mining, where an autonomous mining vehicle is remotely given a mission to move between places A and B within the mine and perform fully automated ore hauling. To avoid physical injuries and collisions with obstacles, the vehicle control system and mission should automatically adapt if people or

The associate editor coordinating the review of this manuscript and approving it for publication was Sotirios Goudos.

obstacles are detected on route. In this article, we evaluate the potential technological enablers for such situation-aware applications within edge computing environments. Next Generation Mining (NGMining)-project [4] aims at facilitating safe and efficient underground mining with autonomous connected and moving mining machinery. Our goal in the NGMining-project was to design and develop an autonomous cognitive robot platform for demonstration purposes. Particularly, the aim was to design and realize scenarios, where an autonomous moving robot detects objects and recognizes human poses, which may be utilized for in situ intelligent decision making and operation of the robot. The challenge was to realize the functionality by using edge computing devices with limited computing power.

Especially, the research was motivated by:

- The lack of suitable models or training data sets for detection of application-specific concepts (i.e., objects and human poses) for the underground mining environment.
- The difficulty of decision making on technology selection for realizing object detection ([5], [6], [7], [8], [9], and [10]) and human pose recognition ([11], [12], [13], and [14]) on edge computing devices with heterogeneous resource-constraints (e.g., different device variants of NVIDIA's Jetson devices).
- Lack of open source-based solution and architecture simultaneously realizing object detection and human pose detection functionalities.

The research question was: How feasible and efficient are object detection and human pose recognition technologies for enabling situation-awareness of robotics applications in edge computing environment? The research was conducted by applying the design science research methodology (DSRM) [15]. The objective was to develop ML-based solution and architecture for facilitating robot operation in edge computing environment with object detection and human pose recognition service.

The contributions of our research are:

- Feasibility and efficiency evaluation of object detection and human pose recognition technologies for edge computing environment. Especially, integration of Yolov5-based object detection [6] and Movenet-based human pose recognition [13] with object concept and human pose concept activation algorithms (executed with Jetson AGX Xavier) may be considered as a novel contribution.
- Additionally, architecture design of the realized solutions in multiple computing environments is presented, which can be considered as a partial evaluation of the ML-based big data reference architecture (RA) [16].

The document is structured as follows: In Section II a literature review is presented covering object detection and human pose recognition technologies, and ML-based big data reference architectures. Research method and process are presented in Section III. Design science-based research iterations on ML for human pose recognition, object detection, and their integration are described in Sections IV-VII. Particularly, the experimented architecture design is presented, evaluated, and compared to related work. Finally, the results are discussed (Section VIII) and concluded (Section IX). The Appendix contains initial experimentation results on object detection and human pose recognition (Iteration 1). Additionally, application programming interface (API) description of object and human pose concepts for the end user is presented. Finally, edge computing devices' SW configurations have been included, which were essential for execution of the experiments.

II. LITERATURE REVIEW

The literature of vision-based scene understanding in human-robot collaboration [17] can be broadly categorized into object perception, human recognition, environment parsing, and visual reasoning. Specifically, we focused on object detection and localization (as part of object perception [17] research area), and human body pose recognition (part of human recognition [17] research area).

Several Yolo-based approaches [5], [6], [7], [8], and [18] have been developed for object detection purposes. Yolov6 [7] improves on earlier approaches with a new neural network design, and training techniques (self-distillation, improved quantization). Yolov7 [8] applied new training techniques (e.g., new model re-parameterization and dynamic label assignment method) for improving accuracy without affecting inference latency. However, Yolov6 achieved higher processing performance in comparison to Yolov7 [7]. A selective frame-down sampling method has been developed for improving Yolo-based object detection (people counting application) performance with Jetson Nano-devices [10]. Particularly, Yolov5 models optimized with TensorRT achieved ~63-108 ms inference latency. Yolov4 performance with optimized models (TF-Lite, TensorRT) has been experimented with Jetson Xavier-platform [9]. The models optimized with TensorRT achieved ~18-62 ms inference latency with COCO and VOC data sets [9].

Partitioning of Yolov2-model between cloud (virtual machine with Tesla P4 GPU) and edge computing resources (Raspberry Piv3) has been studied for improving processing performance [19]. The results indicated that a partitioned model achieved lower performance when compared to edge/cloud-only execution. Additionally, general approaches for deep neural network (DNN) partitioning into a small head-part (executed in edge-devices) and a large tail-part (executed in cloud domain) are available ([20] and [21]). BottleFit [21] achieved 55-89 % lower latency in split CNN computing vs. cloud/edge-only execution approaches with Jetson Nano and Raspberry Piv4 devices. By using a similar approach, DNNs were split between edge (Jetson TX2 and NVIDIA Drive PX2 devices) and cloud domains [22]. Finally, memory efficient patch-based inference for microcontrollers (with only hundreds KBs of RAM) has been proposed [23], which reduces peak memory consumption of existing models by 4-8x.

Google has published several convolutional neural networks (CNNs) in Tensorflow Hub for human joint keypoint identification [13]. Models supporting single and multiple human keypoint identification (up to six humans) are available. The identified locations of human keypoints can be further used for training human pose recognition models [24]. Accuracy of Movenet was compared to other models (OpenPose, PoseNet, MediaPipe Pose) [11]. Movenet had the best performance in terms of percentage of detected joints in figures and videos [11].

Many additional approaches have been studied for realizing human pose recognition. LitePose [25] is an efficient single-branch solution, which has achieved low prediction latency (22-97 ms) with Jetson Nano. A lightweight model based on shrinking deconvolution via Darkpose, and distillation training achieved ∼16 ms inference latency with COCO image dataset on Jetson Xavier [26]. OpenPose-based human body joint prediction integrated with CNN-based human posture (e.g., fall) recognition achieved 2.5 frames per second (FPS) on Jetson Nano [14]. Yolo has been enhanced for multi-person human pose recognition using object keypoint similarity loss [27]. PoseTed [12] utilized Yolov4 for human recognition, and spatial transformer network (STN) as a cropping filter for images and bounding boxes. Then, a feed-forward network predicted human keypoints based on features provided by a transformer encoder/decoder network. PoseNet-based human pose recognition has been integrated as part of a collision-free human-robot collaboration system [28]. Finally, datasets have been published for training human pose classification models. Hierarchical data set (Yoga82) [29] has been used for classification of yoga poses. MPII Human Pose dataset [30] includes 410 human activities in 25 K images covering over 40 K persons.

In addition to technology development, many architectures of big data systems have been realized and published, which increasingly focus on the utilization of ML in edge computing systems [16]. Reference architectures ([31] and [16]) have been developed for facilitating the design of concrete architectures, and communication between important stakeholders. Reduced development and maintenance costs of systems, and reduced risks [32] are additional benefits of RAs. The learning curve associated with RA adoption [32] is a drawback of RAs. Despite the development of RAs, only a few of the RA proposals have been evaluated [16], [31], [33], and [34] (at least partly) based on real-world implementations of big data systems.

The review indicates that many Yolo-based [5], [6], [7], and [8] object detection approaches have been developed. When targeting model execution on edge computing devices with limited resources, the models may be optimized/compressed (TF-Lite, TensorRT) [9] and [10] or partitioned [19], [20], [21], and [22] between edge and cloud environments. Many solutions [12], [27], and [28] have been developed for human pose recognition, also targeting execution on mobile devices [13] and [14]. Human pose data sets [29] and [30] are available, which may be utilized for ML-based model development. Thus, technologies and data sets are available for designing and realizing an architecture, which may facilitate object detection and human pose recognition in edge computing environment.

III. RESEARCH METHOD AND PROCESS

The research was conducted by applying the DSRM [15] (Fig. 1). Initially, there was a need for building a robot-platform/architecture, which has an ability to detect objects and recognize human poses in the context of underground mining. Our objective was to develop a ML-based solution and architecture, which provides a service for facilitating object detection and human pose recognition in edge computing environment. Five research iterations were realized where ML-based solution and architecture was designed and implemented, demonstrated, and evaluated. Especially, the enabling ML-based solutions (technologies) were evaluated in terms of feasibility and efficiency. Additionally, the realized architecture design has been mapped into the architectural elements of the RA [16], which is important for ensuring empirical validity of the RA (step 6 in empirical RA evaluation [35]). Finally, a big data architecture (designed based on the RA) is presented enabling object detection and human pose recognition in edge computing environment.

Videos and images of human poses were used in the research. Informed consent of the involved human (the corresponding author) was obtained for the research.

IV. DESIGN, DEVELOPMENT, AND EVALUATION: HUMAN POSE RECOGNITION (ITERATION 2)

A ML-based architecture was designed and implemented for human pose recognition. Movenet-based [24] human pose recognition technology was chosen as the core ML-based method for further development after initial experimentation (see iteration 1 in the Appendix).

A. ARCHITECTURE

The architecture has been designed based on a Reference Architecture for Big Data systems [16]. We utilized the deployment environment view of the RA for architecture design. Notation of the view is comprised of functionalities (described with rectangles), data stores (ellipsis), and data flows (arrows). The X-axis represents execution of architectural elements in different deployment environments (in-device computing, private cloud, public cloud etc.). The Y-axis divides the elements into functional areas comprising a big data pipeline, where data typically flows from data sources towards the applications (upwards).

Fig. 2 illustrates the ML-based architecture for human pose recognition. Initially, the MPII Human Pose Dataset [30] was used as a data source for training a ML-based human

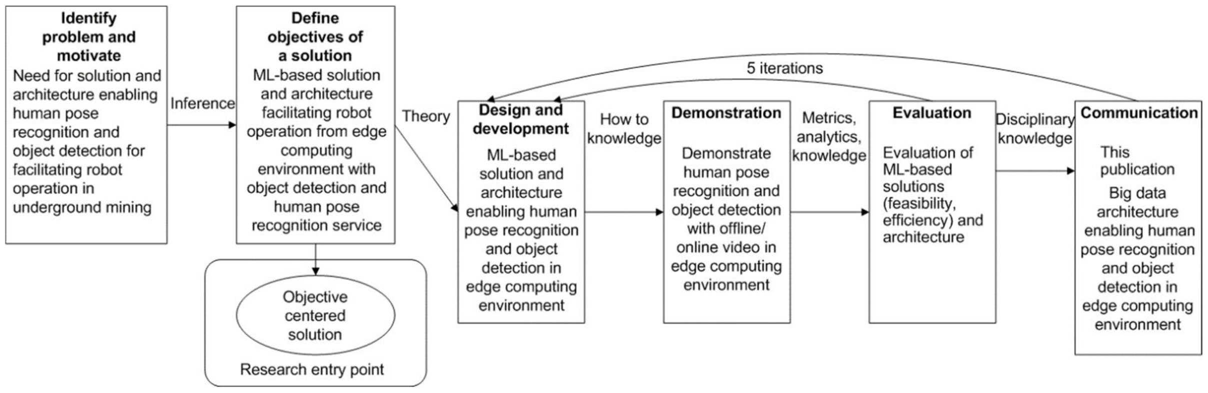
Identify problem and motivate
Need for solution and architecture enabling human pose recognition and object detection for facilitating robot operation in underground mining
Inference
Define objectives of a solution
ML-based solution and architecture facilitating robot operation from edge computing environment with object detection and human pose recognition service
Objective centered solution
Research entry point
Theory
Design and development
ML-based solution and architecture enabling human pose recognition and object detection in edge computing environment
How to knowledge
Demonstration
Demonstrate human pose recognition and object detection with offline/ online video in edge computing environment
Metrics, analytics, knowledge
Evaluation
Evaluation of ML-based solutions (feasibility, efficiency) and architecture
Disciplinary knowledge
Communication
This publication
Big data architecture enabling human pose recognition and object detection in edge computing environment
5 iterations

FIGURE 1. Our research was conducted by applying the DSRM [15].

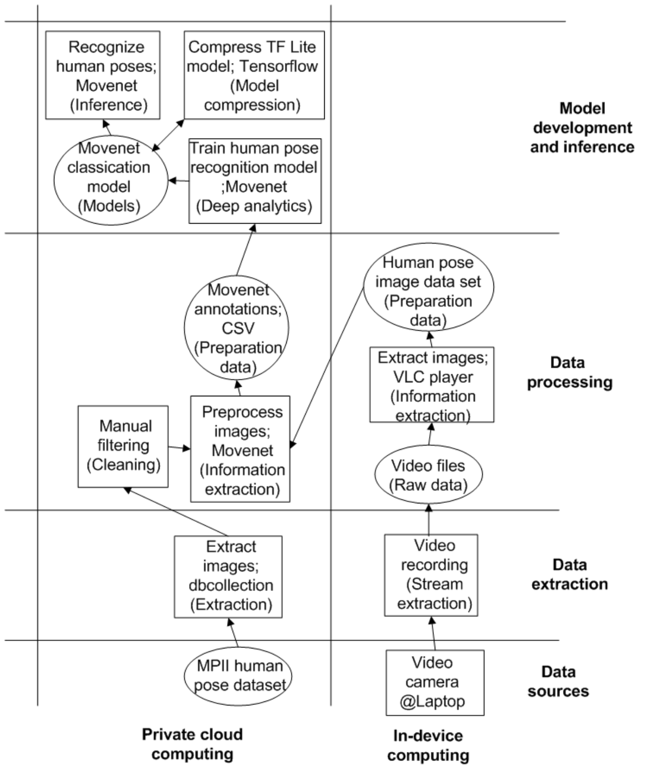
Recognize human poses; Movenet (Inference)
Compress TF Lite model; Tensorflow (Model compression)
Movenet classication model (Models)
Train human pose recognition model ;Movenet (Deep analytics)
Model development and inference
Movenet annotations; CSV (Preparation data)
Human pose image data set (Preparation data)
Extract images; VLC player (Information extraction)
Manual filtering (Cleaning)
Preprocess images; Movenet (Information extraction)
Video files (Raw data)
Data processing
Extract images; dbcollection (Extraction)
Video recording (Stream extraction)
Data extraction
MPII human pose dataset
Video camera @Laptop
Data sources
Private cloud computing
In-device computing

FIGURE 2. ML-based architecture for human pose recognition. Parentheses in architectural elements contain corresponding abstract element in the RA.

pose recognition model. The dataset was downloaded into a private cloud domain for model development purposes. A new SW-component was developed for the extraction of images related to specific categories and activities in the MPII dataset (dbcollection-library [36] was used). Subsequently, the extracted images were reviewed manually, and suitable images were chosen for model training purposes.

However, the MPII Dataset did not include suitable data for modeling all human poses of interest. Therefore, an additional dataset of human poses was created by manually recording video with a camera (i.e., with a laptop). Human pose images were extracted from the video files with VLC video player [37].

The human pose images extracted from the MPII dataset, and the manually collected images were pre-processed by the Movenet-prediction model [13]. The location of human keypoints in images were extracted and stored into comma separated value (CSV)-format. A model was trained based on the CSV-file for human pose classification. Finally, the trained model was compressed into Tensorflow Lite format, which was utilized for inference.

TABLE 1. Collected dataset for human pose recognition.

Human pose	Image count
circle	385
cross	505
handsup	411
left_hand	446
right_hand	406
tform	428
general_total	1021
general_running	*77 (MPII)*
general_walking	*62 (MPII)*
general_standing	*61 (MPII)*
general_right_handup	*399*
general_handup	*421*

B. EVALUATION

1) EFFICIENCY EVALUATION

Table 1 presents the collected data set, which was utilized for training a model for human pose recognition. The size of the data set was 3807 images. The images were divided into seven human pose classes. The general class contained human poses, which were difficult to be distinguished from each other (in italics in Table 1).

The data set was divided into training, validation and testing data with 80 %/10 %/10 % split. A simple neural network architecture was trained, which accepted seventeen human keypoints as input, and predicted seven human poses (see [24] for the detailed implementation). Each human keypoint included X and Y coordinates, and a confidence score [13]. Thus, the size of input data was 51(17*3).

The accuracy of the trained model was 92.7 %. The trained Tensorflow-model (252 KB model file size) achieved 380 ms inference latency with Jetson Nano (4 GB RAM), which was insufficient for our purposes. The Tensorflow-model was compressed into Tensorflow Lite format (27 KB model file size), which achieved 89 % accuracy, and ~2-3 ms inference latency. Despite the slight accuracy decrease, the compressed model is more suitable for inference purposes in edge computing devices.

2) FEASIBILITY EVALUATION

The following lessons were learned during the development:

- Extraction of images from the MPII dataset was not straightforward. dbcollection-library [36] had to be used for extracting images per category from the data set.
- The quality of images in the MPII data set varied, and some of the images had to be manually filtered out.
- Initial model training experiences indicated that similar poses (e.g., walking, running, standing) were difficult to be recognized from each other. Thus, such poses were included under the general class.
- The trained Tensorflow Lite model was tested and evaluated by using video files (including human poses), which were not used for training of the model. Visual analysis of the predicted human poses indicated that most human poses were identified correctly. However, there was confusion between some of the human pose classes. For example, handsup-human pose was sometimes confused with the general-class (which also includes handup-human pose as a sub-category).

3) COMPARISON TO RELATED WORK

We utilized the Movenet-model [13] for recognizing human keypoints, and adapted the implementation [24] for training a human pose classifier. We used some data from the MPII Human Pose dataset [30] for training of the classifier (Table 1). Additionally, we collected new images for identifying custom human poses (Table 1). The combined latency of the compressed (TF-Lite) human pose classification model, and the Movenet-model was much lower (Table 8), when compared to an alternative solution [38] on Jetson Nano-device. Accuracy of the trained and compressed human pose classification model against offline testing data was 89 % (un-compressed model accuracy was 92.7 %). In comparison, PoseNet was trained for recognition of four human poses (right handling, holding, stop, and left handling) and it was integrated with a human-robot collaboration system [28]. We trained a classification model for recognition of 7 human poses, which may explain differences in recognition accuracy (our un-compressed model accuracy was 92.7 % vs. ~100 % [28]). Our work focused on experimentation with edge computing devices, whereas the PoseNet-experiments were performed in desktop PC environment [28].

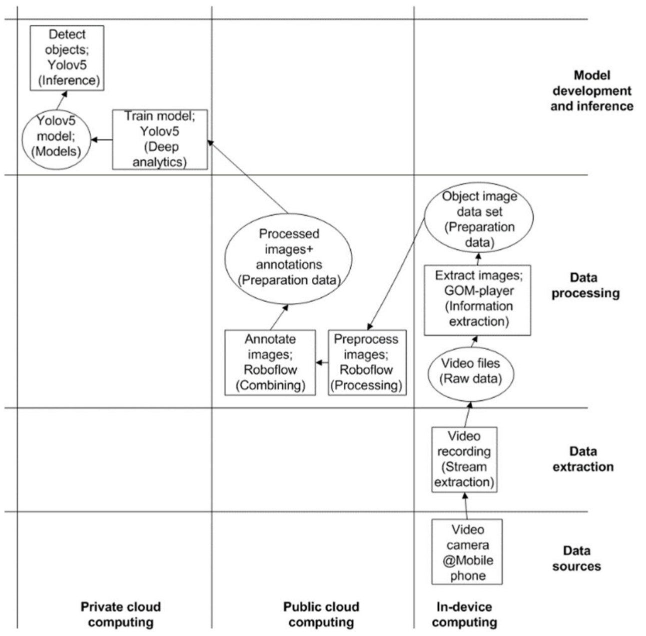
Detect objects; Yolov5 (Inference)
Yolov5 model; (Models)
Train model; Yolov5 (Deep analytics)
Model development and inference
Processed images+ annotations (Preparation data)
Object image data set (Preparation data)
Extract images; GOM-player (Information extraction)
Data processing
Annotate images; Roboflow (Combining)
Preprocess images; Roboflow (Processing)
Video files (Raw data)
Video recording (Stream extraction)
Data extraction
Video camera @Mobile phone
Data sources
Private cloud computing
Public cloud computing
In-device computing

FIGURE 3. ML-based architecture for object detection. Parentheses in architectural elements contain corresponding abstract element in the RA [16].

V. DESIGN, DEVELOPMENT, AND EVALUATION: OBJECT DETECTION (ITERATION 3)

A ML-based architecture was designed and implemented for object detection. Yolov5 was used as the core technology for further development based on initial experimentation (see the Appendix).

A. ARCHITECTURE

Fig. 3 presents the ML-based architecture for object detection, which was designed based on the deployment environment view of the RA [16] (see Section IV-A.1 for a summary of the notation). Three computing environments were identified. Data collection was conducted with a mobile phone (in-device computing [16]), and the collected data set was annotated within the public cloud domain. The object prediction model was trained within the private cloud domain.

Two objects were of interest for our purposes: a robot, and a simulated rock (cardboard box). New data was collected for training an object detection model (based on Yolov5), because the existing models did not contain our custom classes (robot, simulated rock) of interest. Video camera of a mobile phone (Samsung Galaxy A53) was utilized for capturing video files (720p, 30 FPS) of the objects from low/medium/elevated position in relation to the object of interest. GOM-player [39] was used for extraction of JPG-images from the video files. The data set was annotated manually with the Roboflow public cloud service [40]. Particularly, Roboflow used auto-orientation, and resized the images to 640 × 640 format. Subsequently, objects in the images were manually annotated by utilizing the Roboflow user interface (web browser) with a laptop (HP Elitebook 840 G7). Finally,

TABLE 2. Collected dataset for object detection.

	Training data	Validation data	Testing data	Total
Robot	616	171	101	888
Rock	290	81	45	416
Total	906	252	146	1304

a ZIP-file was downloaded into the private cloud domain. The ZIP-file contained the processed images and annotations in Yolo-format. Within the private cloud the images and object annotations were used for training of the object detection model, which was used for inference.

B. EVALUATION

1) FEASIBILITY EVALUATION

The following lessons were learnt during implementation of the architecture:

- The Roboflow service became unresponsive at times during annotation. Therefore, the web browser had to be restarted to improve responsiveness. Additionally, the images were annotated in several batches.
- The Yolov5-model was trained with a GPU (Tesla P100 [41]) with 3584 CUDA-cores and 16 GB of memory. Thus, the GPU was more powerful, when compared to the resources of edge computing devices. Model training took ∼22 minutes. Batch size=64 was used in training, because the model with a larger batch size (128) did not fit into the memory of the GPU.
- Quality of the trained model was evaluated visually by using Yolov5 for inferring objects from previously unseen video files (not used in training). Both objects were successfully detected from the video files, although the probability of prediction was at times lower for the simulated rock.

2) EFFICIENCY EVALUATION

The data set used for training of the object detection model has been described in Table 2. The data set was split between training, validation, and testing with a 70 %/20 %/10 %-ratio.

The model was trained for one hundred epochs with a batch size of sixty-four. The small Yolov5 weights (yolov5s.pt) were used as a starting point for training (transfer learning). Fig. 4 illustrates the training statistics, which were produced by Yolov5. Our medium average precision (mAP_0.5:0.95) and loss metrics seem to improve until one hundred epochs, when the training was stopped. mAP-metrics measure how well the predicted bounding boxes overlap with the ground truth (Intersection over Union (IoU) [42]: 50-95 %). Our mAP statistics of the trained model were: mAP_0.5: 0.99497; mAP_0.5:0.95: 0.88954 (see [42] for a description of the statistical metrics). Thus, the metrics indicate that our trained model can be used for accurate prediction of objects.

VI. DESIGN, DEVELOPMENT, AND EVALUATION: INTEGRATION OF HUMAN POSE RECOGNITION AND OBJECT DETECTION (ITERATION 4)

A ML-based architecture was designed and implemented for executing both object detection and human pose recognition in the same edge computing device.

A. ARCHITECTURE

1) ML-BASED BIG DATA ARCHITECTURE

Initially, the Yolov5-model and the Movenet-model, which were trained earlier (Fig. 2 and 3) in the private cloud environment, were transferred, and loaded into memory of Yolov5 and Movenet-implementations (Fig. 5).

Video was streamed from a camera (Intel RealSense D415) [43], which was connected via a USB-cable to Jetson AGX Xavier-edge computing device (16 GB RAM) [44]. Yolov5 was utilized for video stream extraction (enabled with OpenCV-library [45]). The video stream was transferred internally from Yolov5 to the Movenet-implementation with Python's multiprocessing library [46]. Thus, the same live video stream was processed at both components. Finally, detected objects and recognized human poses were communicated via a Message Queuing Telemetry Transport (MQTT)-broker (Mosquitto [47]) for interested subscribers.

2) INTEGRATION OF OBJECT DETECTION AND HUMAN POSE RECOGNITION

The functionality is illustrated in detail with the unified modeling language (UML) sequence diagram (Fig. 6). The steps are as follows:

- Steps 1-4: Human pose recognition initializes the Movenet detector (human keypoint detection) and classifier (human pose recognition), and the pre-trained models are read into memory.
- Steps 5-6: An internal gesture_processor-class is initialized, which connects to the MQTT-broker.
- Steps 7-8: Yolov5 initializes an internal object_detector-class, which connects to the MQTT-broker.
- Step 9: Yolov5 connects to the Movenet-implementation.
- Step 10: Yolov5 model is loaded into memory.
- Steps 11-13: Video frames are read from the camera. The frames are processed internally and transferred to the Movenet-implementation.
- Step 14: Objects are detected/inferred from the received video frames.
- Steps 15-16: The detected objects and associated probabilities are processed internally (see Fig. 7 for details). When an object is detected, the event is indicated/published to the MQTT-broker.
- Steps 17-18: At the Movenet-implementation, the video frame is provided as a tensor to the Movenet-model.

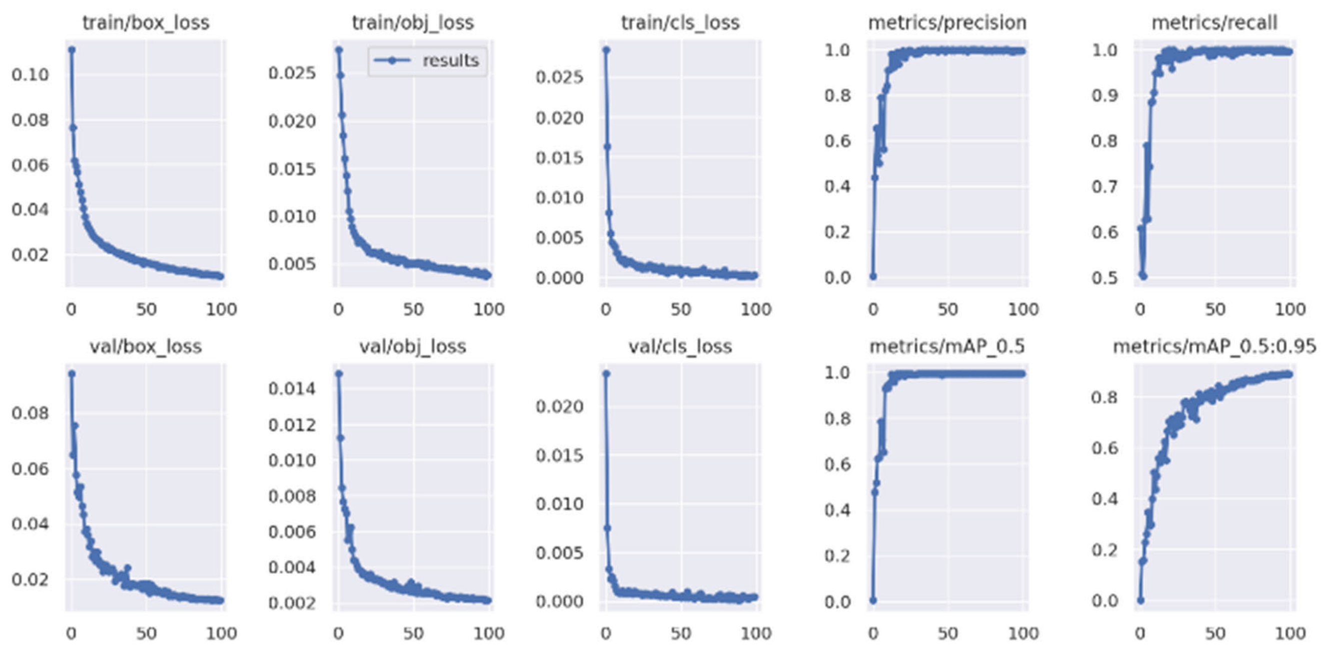

FIGURE 4. Yolov5-model training statistics. The reader is referred to [42] for a detailed description of the statistical metrics.

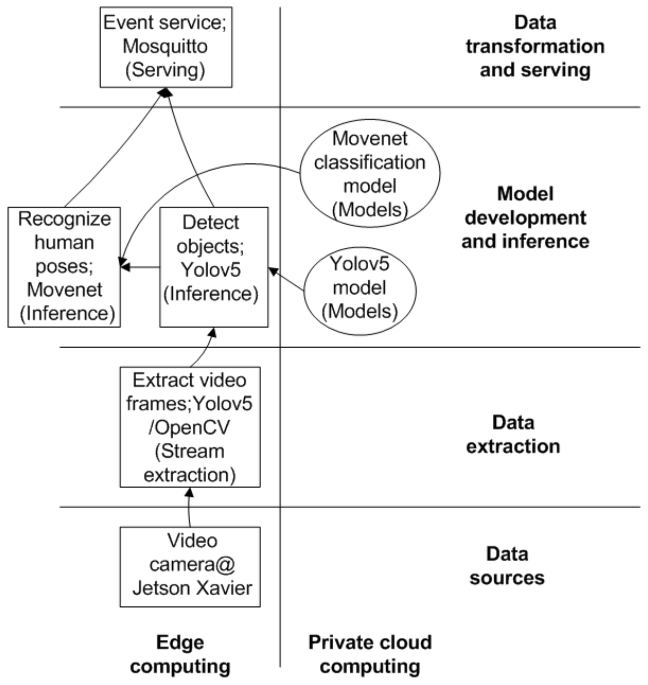
Event service; Mosquitto (Serving)
Data transformation and serving
Movenet classification model (Models)
Recognize human poses; Movenet (Inference)
Detect objects; Yolov5 (Inference)
Yolov5 model (Models)
Model development and inference
Extract video frames;Yolov5 /OpenCV (Stream extraction)
Data extraction
Video camera@ Jetson Xavier
Data sources
Edge computing
Private cloud computing

FIGURE 5. ML-based architecture for execution of object detection and human pose recognition in the same edge computing device.

Keypoints of detected humans are provided as output (as a coordinate list).

- Steps 19-20: For each human, the coordinate list is provided as input to the human pose classifier. The classified human pose and probability are received as output.
- Steps 21-22: The human pose and the associated probability are processed internally (see Fig. 8 for details). When a human pose is recognized, the event is indicated/published to the MQTT-broker.

3) HUMAN POSE CONCEPT ACTIVATION ALGORITHM

Human poses are inferred in each video frame with a different probability. Such real time information may be difficult to be utilized directly for control and decision making of robotics applications. Thus, activation of a higher-level human pose concept was needed. The functionality has been illustrated below with pre-defined constants, and a UML state diagram (Fig. 7). Fig. 7 illustrates how human pose concepts are activated based on the inferred ML-based human poses in video frames. Each ML-based inference (per video frame) is comprised of the detected human pose and associated probability. If probability of the human pose is higher than a pre-defined threshold (trigger_threshold_pose_probability), the pose will be considered for further processing. Additionally, only after enough consecutive occurrences of human poses have been detected (trigger_threshold_consecutive_poses), a human pose concept is activated (POSE_OBSERVED), and the detected pose is stored (current_pose in Fig. 7) The described functionality has been implemented into human gesture_processor-class (Fig. 6).

Pre-defined constants (comments after hashtags):
```
# Probability threshold of recognized poses.
trigger_threshold_pose_probability = 0.9
```

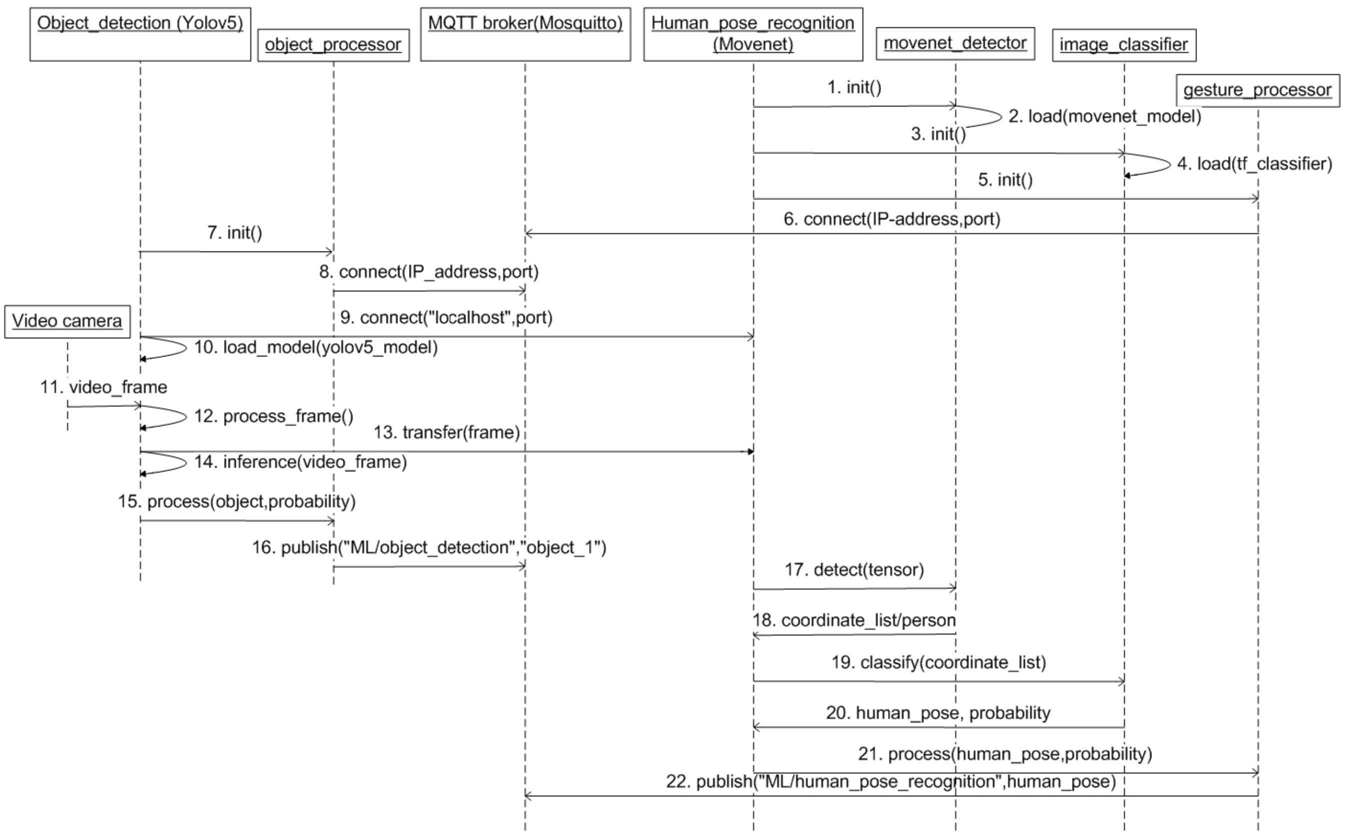
Object_detection (Yolov5)
object_processor
MQTT broker(Mosquitto)
Human_pose_recognition (Movenet)
movenet_detector
image_classifier
gesture_processor
Video camera
1. init()
2. load(movenet_model)
3. init()
4. load(tf_classifier)
5. init()
6. connect(IP-address,port)
7. init()
8. connect(IP_address,port)
9. connect("localhost",port)
10. load_model(yolov5_model)
11. video_frame
12. process_frame()
13. transfer(frame)
14. inference(video_frame)
15. process(object,probability)
16. publish("ML/object_detection","object_1")
17. detect(tensor)
18. coordinate_list/person
19. classify(coordinate_list)
20. human_pose, probability
21. process(human_pose,probability)
22. publish("ML/human_pose_recognition",human_pose)

FIGURE 6. **A sequence diagram illustrating the execution of object detection and human pose recognition on the same edge computing device.**

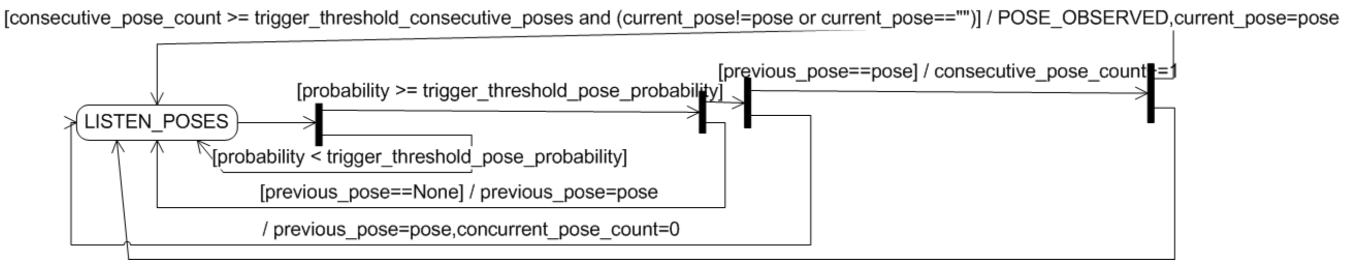
[consecutive_pose_count >= trigger_threshold_consecutive_poses and (current_pose!=pose or current_pose=="")] / POSE_OBSERVED,current_pose=pose
LISTEN_POSES
[probability >= trigger_threshold_pose_probability]
[previous_pose==pose] / consecutive_pose_count+=1
[probability < trigger_threshold_pose_probability]
[previous_pose==None] / previous_pose=pose
/ previous_pose=pose,concurrent_pose_count=0

FIGURE 7. **UML state diagram of human pose concept activation algorithm.**

```
# Threshold of poses in consecutive video frames needed
# for human pose concept activation.
trigger_threshold_consecutive_poses = 4
```

4) OBJECT CONCEPT ACTIVATION ALGORITHM

Additionally, an object concept activation algorithm has been designed and implemented (see Fig. 8 for a simplified view of the implementation). Yolov5 provides either object detections (and associated probabilities) or indications, that no objects were detected in a video frame. If an object has been detected with a probability, which exceeds a pre-defined level of confidence (trigger_threshold_object_probability), a timestamp will be associated with the object (to be used eventually for expiration). Subsequently, if the object has been detected earlier (object_exists==True), an object counter is increased. When enough consecutive object detections have been observed (trigger_threshold_consecutive_objects), an object concept is activated (OBJECT_DETECTED).

If no objects have been detected for a pre-defined threshold (trigger_threshold_no_detections) of frames, all internally saved/activated object concepts will be deleted/cleared (clear_obj_detections()).

Additionally, if an object is detected with a probability, which is lower than a pre-defined threshold (trigger_threshold_object_drop_probability), and enough consecutive object detections with a low confidence have been observed (trigger_threshold_consecutive_drop_objects), the internally saved/activated object concepts will be

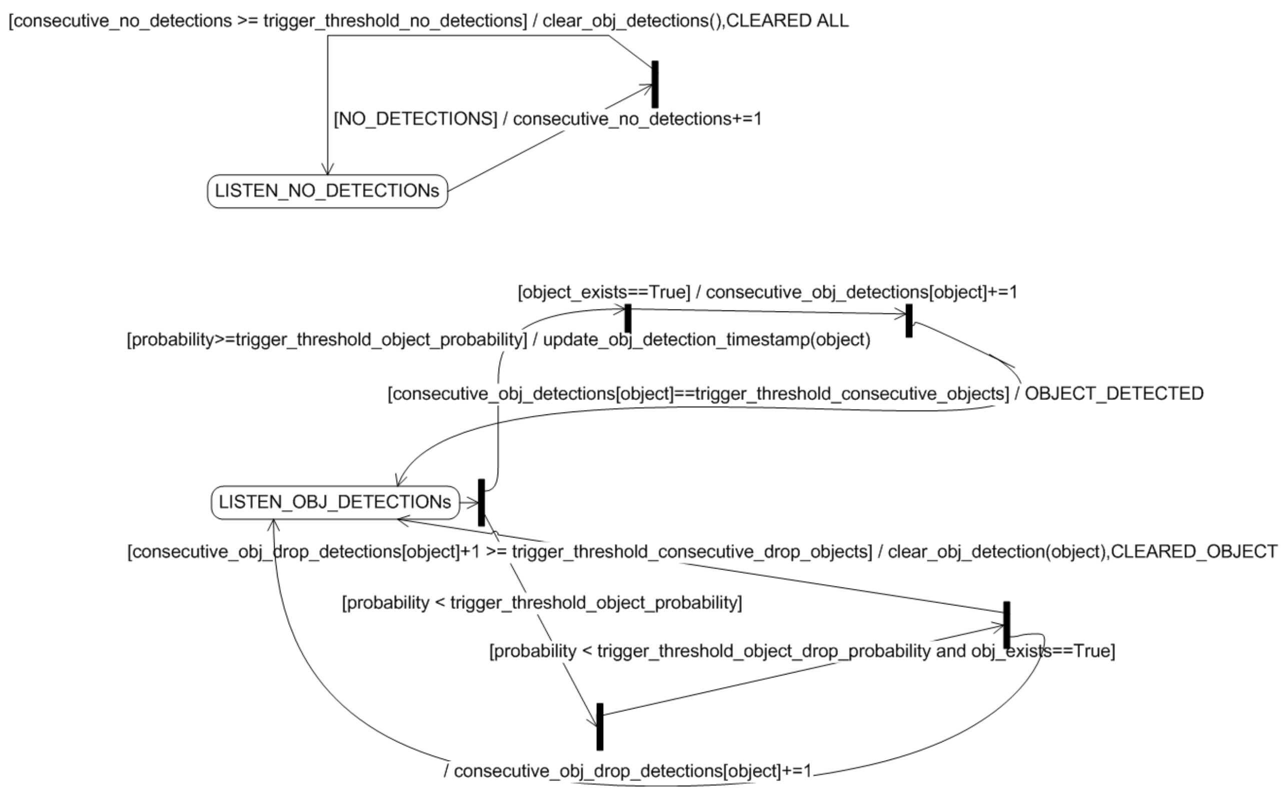
[consecutive_no_detections >= trigger_threshold_no_detections] / clear_obj_detections(),CLEARED ALL
[NO_DETECTIONS] / consecutive_no_detections+=1
LISTEN_NO_DETECTIONs
[object_exists==True] / consecutive_obj_detections[object]+=1
[probability>=trigger_threshold_object_probability] / update_obj_detection_timestamp(object)
[consecutive_obj_detections[object]==trigger_threshold_consecutive_objects] / OBJECT_DETECTED
LISTEN_OBJ_DETECTIONs
[consecutive_obj_drop_detections[object]+1 >= trigger_threshold_consecutive_drop_objects] / clear_obj_detection(object),CLEARED_OBJECT
[probability < trigger_threshold_object_probability]
[probability < trigger_threshold_object_drop_probability and obj_exists==True]
/ consecutive_obj_drop_detections[object]+=1

FIGURE 8. **Simplified UML state diagram of object concept activation algorithm.**

deleted/cleared (clear_object_detection(object)). The event is also indicated to end users (CLEARED_OBJECT).

The functionality described in Fig. 8 has been implemented to the object_processor-class in Fig. 6.

Pre-defined constants (comments after hashtags):

# Threshold for consecutive ''no detection''-events in video
# frames to trigger expiration of all detected objects.
trigger_threshold_no_detections = 8
# Probability threshold to trigger object concept activation.
trigger_threshold_object_probability = 0.6
# Threshold for consecutive object detections in video
# frames (with probability >
# trigger_threshold_object_probability) to trigger
# object concept activation.
trigger_threshold_consecutive_objects = 10
# Probability threshold to trigger deactivation of
# object concept.
trigger_threshold_object_drop_probability = 0.3
# Threshold for consecutive object detections in video
# frames (with probability <
# trigger_threshold_object_drop_probability) to trigger
# deactivation of object concept.
trigger_threshold_consecutive_drop_objects = 4

5) INTEGRATED ML-BASED BIG DATA ARCHITECTURE FOR OBJECT DETECTION AND HUMAN POSE RECOGNITION

Fig. 9 illustrates the big data architecture, which enables ML-based object detection and human pose recognition as a service for end-user applications. The view has been created by synthesizing earlier architectural views (Figs. 2-3 and 5), which were realized as parts of the integrated system.

B. EVALUATION

1) FEASIBILITY EVALUATION

The following lessons were learnt, when implementing the architecture:

- The video stream had to be re-transferred from Yolov5 to the Movenet-implementation, because OpenCV API did not allow two processes to read from the video camera at the same time.
- When 720p video was streamed, Yolov5 scaled the video automatically to 736×1280 due to the Yolov5-model backbone stride size (32). Thus, the size of the video frame had to be a multiple of 32.
- As the video stream was transferred synchronously between the processes, the slower process became a bottleneck (Movenet) in terms of processing performance. It would also be possible to transfer a video stream asynchronously. In this case the slower process would lag (this approach was also experimented).
- It was discovered that Movenet-based human pose recognition executed with CPUs had similar performance (in terms of FPS), when compared to execution with a GPU. To save GPU-memory, human pose

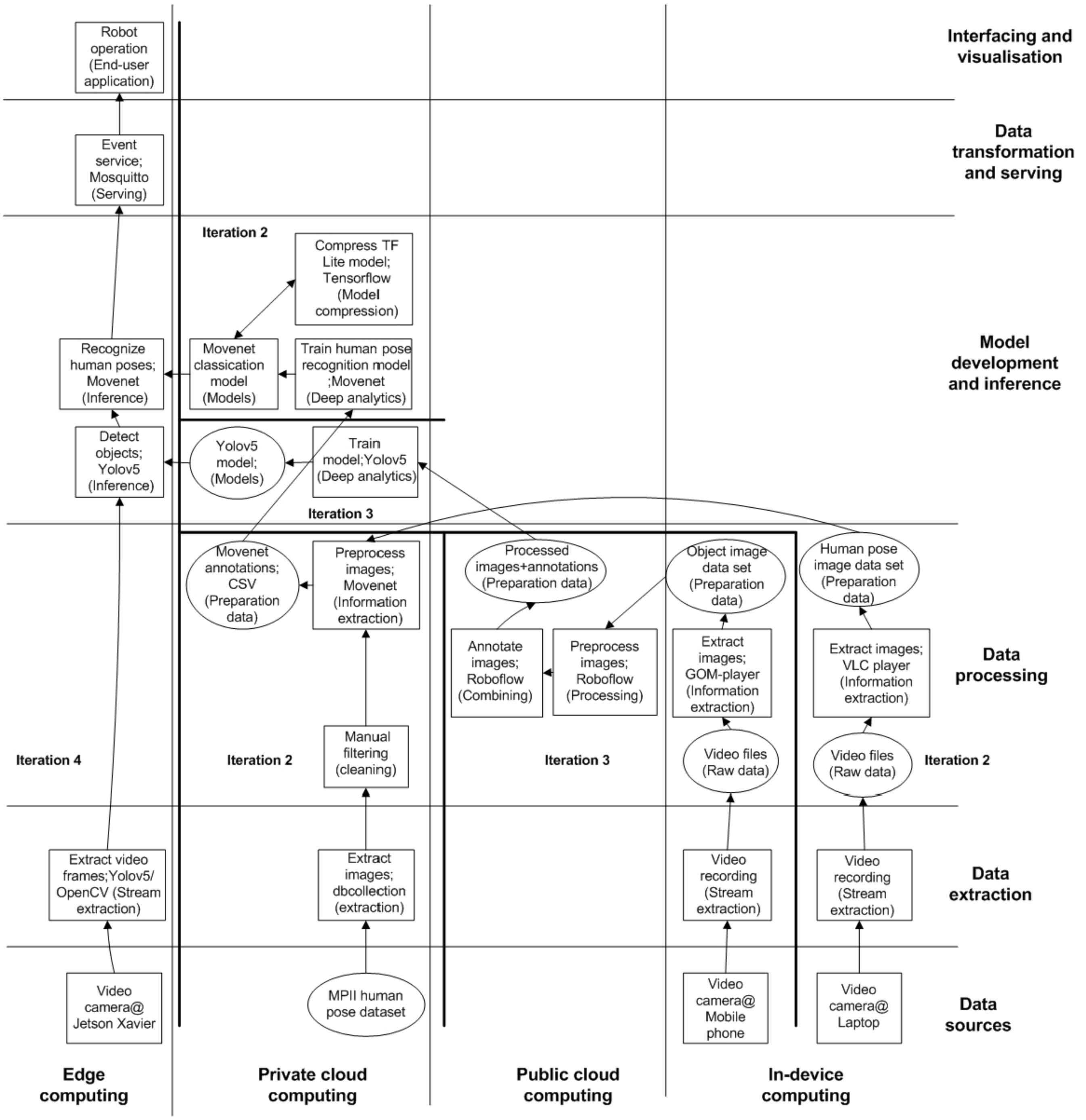
Robot operation (End-user application)
Event service; Mosquitto (Serving)
Interfacing and visualisation
Data transformation and serving
Iteration 2
Compress TF Lite model; Tensorflow (Model compression)
Recognize human poses; Movenet (Inference)
Movenet classication model (Models)
Train human pose recognition model ;Movenet (Deep analytics)
Model development and inference
Detect objects; Yolov5 (Inference)
Yolov5 model; (Models)
Train model;Yolov5 (Deep analytics)
Iteration 3
Movenet annotations; CSV (Preparation data)
Preprocess images; Movenet (Information extraction)
Processed images+annotations (Preparation data)
Object image data set (Preparation data)
Human pose image data set (Preparation data)
Annotate images; Roboflow (Combining)
Preprocess images; Roboflow (Processing)
Extract images; GOM-player (Information extraction)
Extract images; VLC player (Information extraction)
Data processing
Iteration 4
Iteration 2
Manual filtering (cleaning)
Iteration 3
Video files (Raw data)
Video files (Raw data)
Iteration 2
Extract video frames;Yolov5/ OpenCV (Stream extraction)
Extract images; dbcollection (extraction)
Video recording (Stream extraction)
Video recording (Stream extraction)
Data extraction
Video camera@ Jetson Xavier
MPII human pose dataset
Video camera@ Mobile phone
Video camera@ Laptop
Data sources
Edge computing
Private cloud computing
Public cloud computing
In-device computing

FIGURE 9. Big data architecture illustrating ML-based object detection and human pose recognition. Architecture design of different DSR-research iterations has been separated with thick lines.

recognition was forced to be executed only by utilizing CPUs of the device.

- It was apparent, that an abstract human pose recognition and object detection concept (Figs. 7-8) had to be implemented to increase usability of the provided API (see the Appendix). The functionality enables expiration and filtering of detections/recognitions associated with impermanence or low confidence.

2) EFFICIENCY EVALUATION

First, efficiency of human pose recognition was experimented by streaming video from a file and from the camera. The following parameters were utilized in the experiments:

- Video files: 3-minute video files: no humans (white wall); one human with human poses
- Video resolution: High resolution (720 × 1280); low resolution (480 × 640)

TABLE 3. **Efficiency evaluation results of human pose recognition experiments with Jetson AGX Xavier.**

File	CPU-util.(%)	GPU-util.(%)	RAM (GB)	GPU (GB)	FPS
Low resolution; no humans	40.9	0.0	2.1	0.3	18.6
Low resolution; human poses	37.0	0.0	2.1	0.3	15.3
High resolution; no humans	41.4	0.0	2.2	0.3	20.1
High resolution; human poses	35.3	0.1	2.2	0.3	14.8
Camera					
Low resolution; human poses	35.9	0.0	2.2	0.3	15.4
High resolution; human poses	36.7	0.0	2.2	0.3	16.9

TABLE 4. **Efficiency evaluation results of object detection experiments with Jetson AGX Xavier.**

File	CPU-util. (%)	GPU-util. (%)	RAM (GB)	GPU (GB)	FPS
Low resolution; no objects	17.3	70.6	5.1	1.1	44.8
Low resolution; objects	17.2	61.8	5.1	1.1	38.3
High resolution; no objects	15.0	77.9	5.3	1.1	19.8
High resolution; objects	14.8	72.9	5.2	1.1	18.3
Camera					
Low resolution; objects	17.1	63.3	6.2	2.1	38.2
High resolution; objects	13.8	73.3	6.7	2.5	17.5

- Video stream from a camera: no humans (white wall); one human with human poses; 30 FPS

The following measurements were performed:

- FPS was measured as an average across all the processed video frames.
- GPU/CPU memory consumption and CPU/GPU utilization was measured with jetson_stats-library [48]. The measurement tool recorded measurements every 0.5 s during the experiment. Additionally, CPU-utilization was calculated as an average across all CPUs (8).

a: HUMAN POSE RECOGNITION EXPERIMENTS

Efficiency evaluation results of human pose recognition experiments are presented in Table 3. Both CPU memory consumption (2.2 GB/16 GB) and CPU-utilization (35-41 %) remained at a low level. When human poses are recognized, performance decreases a few FPS to ~15-17 FPS. There is only a slight difference (~0.5-1.5 FPS) in the processing performance when different video resolutions are utilized in human pose recognition.

b: OBJECT DETECTION EXPERIMENTS

Efficiency of object detection was experimented by streaming video from a file and from the camera. The following parameters were utilized in the experiments:

- Video file: 3-minute video files: no objects (white wall); two objects
- Video resolution: High resolution (720 × 1280); low resolution (480 × 640)
- Video stream from a camera: no objects (white wall); two objects; 30 FPS

Measurements were performed similarly as the human pose recognition experiments. Efficiency evaluation results of object detection experiments are presented in Table 4.

GPU-utilization was quite high (~62-78 %). However, CPU utilization stayed at a low level (~15-17 %). Total RAM (5-7 GB/16 GB) and GPU memory consumption (1.1-2.5 GB/8 GB) was low. There was a slight drop in processing efficiency (~1.5-5.5 FPS) when objects were inferred. When video is streamed from a video camera, more GPU-memory is utilized (1-1.4 GB), when compared to streaming from a file. There is a significant difference in processing performance (~20-25 FPS) when high resolution video is processed.

c: SIMULTANEOUS HUMAN POSE RECOGNITION AND OBJECT DETECTION EXPERIMENTS

Efficiency of simultaneous object detection and human pose recognition was experimented by streaming video from the camera. The following parameters were utilized in the experiments:

- Video resolution: High resolution (720 × 1280); low resolution (480 × 640)
- Video stream from a camera: 3 minutes video stream with two objects and one human with poses; 30 FPS

The experiments were performed five times. Measurements were performed similarly as the human pose recognition experiments. Efficiency evaluation results of simultaneous object detection and human pose recognition experiments are presented in Table 5. CPU-utilization (~40-44 %) is only slightly higher, when compared to human pose recognition results (Table 3). However, GPU-utilization is lower (~27-59 %), when compared to the object detection results (Table 4). This can be explained by the lower processing performance of human pose recognition, which is a bottleneck in the architecture. Thus, Yolov5 processes video at a suboptimal rate (see Table 4), and GPU utilization is also lower. There is a slight increase (~0.4-0.5 GB) in RAM-usage, when compared to object detection (Table 4). In overall, ~13-16 FPS can be processed while keeping CPU/GPU resource consumption at a low/moderate level.

d: COMPARISON TO RELATED WORK

The integrated ML-based architecture was designed and realized for object detection (Yolov5) and human pose recognition (Movenet). A similar approach has not been

TABLE 5. Efficency evaluation results of simultaneous object detection and human pose recognition experiments with Jetson AGX Xavier.

File	CPU-util. (%)	GPU-util. (%)	RAM (GB)	GPU (GB)	Yolov5: FPS	Movenet: FPS
Low resolution	43.5	26.9	5.6	1.1	15.5	15.5
High resolution	42.6	58.5	5.7	1.1	13.9	13.8
Camera						
Low resolution	42.3	27.2	6.6	2.1	14.9	15.1
High resolution	39.9	58.3	7.1	2.5	13.4	13.7

published earlier according to the authors' best knowledge, where processing performance (FPS) and HW resource consumption (Table 5) has been evaluated on Jetson AGX Xavier-platform. Particularly, new functionality was implemented for integrating Yolov5 and Movenet with the MQTT broker (Fig. 6), for realizing transfer of video camera frames with inter-process communication (Fig. 6), and for abstracting object and human pose concepts (Figs. 7 and 8) to the end user (see the Appendix). Gomes [10] also utilized Python's multiprocessing-library for inter-process communication between Yolov5-related components. However, we used Connection-class for communication between Yolov5 and Movenet-implementations instead of using the Queue-class [10]. Object detection (based on R-CNN) and human pose recognition (based on CNN) were integrated for facilitating skill transfer in manufacturing systems [49]. However, the performance of the models was not experimented simultaneously with edge computing devices in real time, which was part of our contribution.

During the development of the integrated ML-based architecture, we learned that inference latency of the Movenet-model could not be improved with execution on a GPU (vs. execution on CPUs). The Movenet-model [13] was originally developed for consumer-devices (i.e., laptops, cell phones without a powerful GPU), which may explain similar processing performance on CPU/GPU-devices.

In terms of processing efficiency, the time required to detect a new or a changed concept (object/human pose) can also be estimated. In our implementation, concept detection time is dependent on the configuration of constants (trigger_threshold_consecutive_poses, trigger_threshold_consecutive_objects), which are used for determining how often an object/human pose must appear in consecutive video frames (see Fig. 7 and 8). Thus, concept detection time can be calculated based on the aforementioned constant divided by FPS.

The realized ML-based architecture (Fig. 9) may be considered as an evaluation of the deployment environment view of the RA for big data systems [16]. Especially, the realized architectural elements/components were mapped to the abstract elements defined in the RA [16], which is important for ensuring empirical validity of the RA [35]. Also, the architectural elements were placed into different deployment environments and functional areas of a big data pipeline [16]. The deployment environment view of the RA was suitable for designing the implementation architecture, as all the existing elements could be mapped to the RA. The presented architecture (Fig. 9) can be considered as an additional partial evaluation (see other evaluations: [33], [50]) of the deployment environment view of the RA.

VII. DESIGN, DEVELOPMENT, AND EVALUATION: FINAL OBJECT DETECTION EXPERIMENTS (ITERATION 5)

After the initial object detection experiments with Yolov5 (Iteration 1 in the Appendix), a more extensive evaluation was conducted with the most recent technologies. The purpose of the experiments was also to evaluate, if object detection without human pose recognition could be realized with a more resource-constrained edge computing device (Jetson Nano [51], 4 GB RAM). Also, the goal was to find out the most promising technologies from processing performance point of view.

A. EFFICIENCY EVALUATION

Object detection performance with Jetson Nano was experimented as follows:

- A HD video file (720p; length: 1 min 03 s) was used for object detection.
- The inferred video frames were not visualized nor saved into a file.
- Each experiment was executed five times.
- Processing performance was measured (FPS). All implementations (except Yolov4) were instrumented for FPS measurement with timestamps.

Yolov4 was not instrumented for measurements (C-based implementation). Instead, FPS measurements (for each video frame) reported by Yolov4 was provided as an average. For mcunet [23], the original code (eval_det.py) was modified for enabling video streaming from a file (the original only enabled inference of an image). Additionally, image and bounding box scaling/rescaling was performed between 720p video file (1280×720), and video frame size (160×128), which was used for inference.

The results of the experiments are presented in Fig. 10. Yolov4 [5] achieved the highest processing performance. TensorRT optimized Yolov5-models enabled ~28-42 % higher processing performance, when compared to unoptimized models. TinyML approach of mcunet [23] was the third fastest. Lower precision/reduced model versions (e.g., Yolov5n and Yolo6n) enabled higher performance as expected. The results indicated that object detection may be realized without human pose recognition also with a less powerful edge computing device (Jetson Nano).

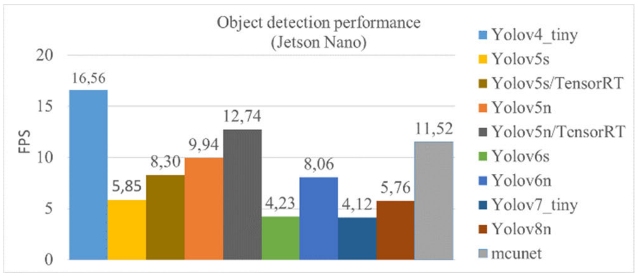

FIGURE 10. Object detection performance results with Jetson Nano.

TABLE 6. Accuracy (AP) and computational complexity(flops) of object detection models (as reported in the literature [7], [8], [52], [53], and [56]).

Model	AP (COCO 2017, validation data) %	FLOPS (B)
Yolov4_tiny	21.7	6.9
Yolov5s	37.4	16.5
Yolov5n	28	4.5
Yolov6s	43.5-44.3	44.2-45.3
Yolov6n	35.9-37	11.1-11.4
Yolov7_tiny	37.4-38.7	13.7-13.8
Yolov8n	37.3	8.7
mcunet	NA	0.168

Accuracy and computational complexity of the experimented models (except mcunet [23]) was evaluated based on the literature (Table 6). The fastest models (Yolov4_tiny, Yolov6n) seem to be associated also with a lower accuracy. Likewise, the most accurate models (Yolov6s, yolov7_tiny) achieved slowest processing performance in our experiments (Fig. 10). Computational complexity has been evaluated based on the number of floating-point operations required per second (as billion FLOPS). The most accurate model (Yolov6s) is associated with the highest computational requirements. Respectively, the least accurate models (Yolov4_tiny, Yolo5n) are associated with lower computational requirements. mcunet has the lowest computational requirements, but accuracy with COCO 2017-dataset has not been reported (to the authors' best knowledge).

B. COMPARISON TO RELATED WORK

The results indicated that Yolov4 was faster than Yolov5 (Fig. 10). We achieved a similar result with Yolov4_tiny executed on Jetson Nano (16.6 FPS), which has been reported in an earlier study (16 FPS) [52]. Thus, Yolov4 could also have been selected for realizing object detection from efficiency point of view. However, Yolov5 has been implemented with Python and Yolov4 has been implemented mostly with C/C++. The modifications needed for integrating Yolov5 with Movenet and MQTT, and for realizing new functionality (Figs. 5 and 6) was easier to the main author with Yolov5/Python. mcunet (TinyML) [23] may alternatively be used for realizing object detection (third fastest processing performance). Finally, TensorRT may be used for improving Yolo performance [9], [10]. We able to improve processing performance by ∼28-42 % by compressing Yolov5-models with TensorRT. Our results are at a similar range, when compared to an earlier study [10], where TensorRT-optimization of Yolov5 was experimented with Jetson Nano (50 % improvement in latency). However, we needed to export TensorRT-models with half-precision (FP16) due to memory constraints of Jetson Nano.

When we compared accuracy of the experimented object detection models (based on the results reported in the literature) to the processing performance, we noticed that the fastest models are typically associated with a lower accuracy and computational requirements (as expected). However, accuracy of the mcunet-model has not been compared to Yolo-based models in related work (with COCO-data set) according to the authors' best knowledge. The tradeoff between processing speed and accuracy should be considered, when making a technology choice for object detection. Mean average precision of our Yolov5-model was close to 99.5 % (mAP_0,5: 0.99497) against offline testing data. The high accuracy may be explained by having only two objects (simulated rock and robot) of interest for modeling purposes.

When models are compressed (e.g., with TensorRT), accuracy may be affected. In earlier experiments, TensorRT-optimized Yolo-based models in edge-devices (Jetson AGX Xavier, Jetson Nano) have led to a small decrease in accuracy [10] or to improved accuracy [9]. As an additional test, we optimized our trained Yolov5-model with TensorRT, and tested accuracy with validation data of our image data set on Jetson Nano. However, there was no difference in accuracy (mAP_0,5:0.995), when compared to the uncompressed Yolov5-model.

VIII. DISCUSSION

Even though the architecture was originally designed for the underground mining context, it may also be applied in other edge computing contexts, where object detection and human pose recognition functionalities need to be realized. Especially, the object concept and human pose concept algorithms can be adapted by configuration of the constants (see Sections VI-A.3-A.4) based on the situation. Additionally, different strategies may be specified (e.g., lowly/highly sensitive configuration of a set of constants) to the algorithms.

We evaluated the computational complexity of object detection models (in Table 6) based on published results. Computational complexity of Movenet-based human pose recognition is mostly dependent on the performance of multiple human keypoint detection (see Table 8 in the Appendix). Multiple human keypoint detection uses MobileNetV2 image feature extractor with Feature Pyramid decoder followed by CenterNet prediction heads [13]. The reported complexity of MobileNetV2 is 300-585 million multiply-add operations/second (depending on version of MobileNetV2) [58].

Applicability of the architecture may be limited by inability to generate a custom data set of objects and human poses for model training purposes. Also, GPU-equipped devices must be available for training of models (Tesla P100 was used), and realizing inference-related (Jetson AGX Xavier was used) functionality. Feasibility evaluation of human pose recognition (Section IV-B2) indicated the limitation in classification of human poses. Especially, the modeled human poses should differ significantly from each other to achieve better accuracy. Finally, sensor-based measurements (e.g., distance sensors) may also be needed for supporting safe and reliable decision-making in autonomous robotics applications due to inaccuracy of ML-based predictions.

Future work may include partitioning of models between edge/cloud deployment environments, which may improve processing performance. Only few of the model partitioning approaches [19], [20], [21], and [22] have been published as open-source solutions [54]. Additionally, local models may be trained by collaborating nodes using a dedicated dataset. In this approach, the local models are used for aggregating a global model. Such federated learning has privacy benefits [55]. Processing performance of human pose recognition may be improved with an alternative modeling approach (e.g., Yolo-based [27] and [12] or LitePose [25]). Additionally, hand gestures may be recognized [57], which can be considered as complementary/replacement to the recognition of human body poses. Another item for future work is further research related to the service API (see API description in the Appendix), which is provided for facilitating decision making of robotics applications. We experimented with a live video camera by visually viewing activated human pose and object concepts provided by the API. However, new research is needed for integrating our study with a robotics application with reliability requirements, which may also increase technology readiness level (TRL) of our solution (current TRL level=4/5). A related future challenge is accuracy of the prediction models in a real underground mining environment.

IX. CONCLUSION

The research question focused on evaluating feasibility and efficiency of object detection and human pose recognition technologies for enabling situation awareness of robotics applications in edge computing environment. Feasibility and efficiency evaluation of Yolov5-based object detection and Movenet-based human pose recognition on Jetson AGX Xavier-platform may be considered as an answer to the research question, and as a new contribution. Object concept and human pose concept activation algorithms may be considered as an additional contribution. Several feasibility related challenges regarding the experimented technologies, datasets, and services had to be solved, which were presented as lessons learnt in the evaluation. In terms of efficiency, Yolov5 and Google's Movenet models enabled simultaneous human pose recognition and object detection on Jetson AGX Xavier edge computing platform with acceptable processing performance (~13-16 FPS). Additionally, GPU-utilization (~27-59 %) and CPU-utilization (~40-44 %) remained at a medium level, and most of the memory remained unused ($<$ 44 % total memory consumption) for other processes. Further object detection experiments on Jetson Nano edge computing device indicated potential for improvements with alternative technologies (Yolov4, mcunet/TinyML) or with TensorRT-based optimizations. Additionally, architecture design of the realized solutions in multiple computing environments can be considered as a partial evaluation of the ML-based big data reference architecture [16].

APPENDIX

A. INITIAL OBJECT DETECTION AND HUMAN POSE RECOGNITION EXPERIMENTS (ITERATION 1)

Object detection and human pose recognition technologies were initially reviewed and selected for further experimentation. The goal was to experiment efficiency of the technologies for further development in edge computing environment.

TABLE 7. Efficiency evaluation results of YOLOv5 object detection technology with Jetson Nano.

	Jetson Nano	Tesla P100
Yolov5	**Inference latency (ms)**	**Inference latency (ms)**
Yolov5n6	84	7
Yolov5n	71	5

1) EVALUATION

a: EFFICIENCY EVALUATION OF OBJECT DETECTION

Table 7 presents efficiency evaluation results, when Yolov5 object detection technology was experimented with Jetson Nano (4 GB RAM) [51] and a virtual machine with Tesla P100-GPU [41]. HD video file (720p) was processed, and processing efficiency was evaluated. Yolov5 achieved adequate processing performance, which led us to continue further experimentation with the technology.

b: EFFICIENCY EVALUATION OF HUMAN POSE RECOGNITION

Table 8 presents efficiency evaluation results, when single/multiple human pose recognition models were experimented with Jetson Nano. A HD video file (720p) was processed, and inference latency was evaluated. Two ML-based models are needed for realizing human pose recognition based on Movenet. First, the human keypoints are detected from the video frame. Subsequently, the predicted human keypoints are input to a human pose classifier. The results indicated that human keypoint detection is significantly more time-consuming, when compared to human pose classification. The smaller (and less accurate) multiple human Movenet-model achieved a lower inference latency.

Additionally, mmpose [38] was experimented for comparison. However, a low processing performance was achieved (0.4 FPS).

TABLE 8. Efficiency evaluation results of human pose recognition models with Jetson Nano.

Model	Inference latency (ms)
Single human keypoint detection (movenet_thunder.tflite)	380
- Pose classifier (TF Lite model)	1
Multiple human keypoint detection (multipose_lightning.tflite)	110
- Pose classifier (TF Lite model)	1

Finally, Yolov5 and Movenet-based multiple human keypoint detection methods were experimented simultaneously with Jetson Nano. The processes competed for the same GPU-resource, and the processing performance of both technologies dropped significantly (to ∼5-7 FPS). Also, sometimes the device became unresponsive, and had to be rebooted. Thus, Jetson Nano did not provide enough processing performance for our goal of integrating object detection and human pose recognition on one device.

B. API DESCRIPTION

When abstract events of objects were detecte and human poses were recognized (as described in sections VI-A3 and VI-A4), the events were notified to subscribers (e.g., robot application) via a MQTT-broker. A separate MQTT-topic was used for indicating different detection events. The MQTT-API for human pose events contained the recognized human pose class (Table 1). The MQTT-API for object detection events:

"detectedobject_identifier", where identifier is an integer (e.g., rock_0).

"cleared detectedobject_identifier", where identifier is an integer (e.g., cleared rock_0).

"cleared all", when all previous objects detections were deleted/cleared.

C. DEVICE SW-CONFIGURATIONS

Table 9 describes SW-configurations of edge computing devices, which were used in the experiments. Jetson Nano needed Python v3.6 for installation of TensorRT (v8.2.1.8). Additionally, Python v3.8 was required for execution of Yolov8. However, TensorRT installation packages were not available for Python v3.8 on Jetson Nano. Thus, two execution environments (Python v3.6/v3.8) were used for execution of the experiments on Jetson Nano. The latest available JetPack (v4.6) enabled support for both Python v3.6 and v3.8, which were needed for creating execution environments for TensorRT/Yolov5 and Yolov8.

TABLE 9. Edge computing devices' important SW-configurations.

	Jetson Xavier	Jetson Nano	Jetson Nano	Jetson Nano
Experimented technologies	**Yolov5, Movenet**	**Yolov4, Yolov5, Yolov6, Yolov7, mcunet, Movenet**	**TensorRT/ Yolov5**	**Yolov8**
JetPack	v4.6	v4.4	v4.6	v4.6
CUDA	v10.2	v10.2	v10.2	v10.2
Tensorflow	v2.31	v2.31	-	-
Torch	v1.8.0	v1.8.0	v1.80	v1.11
Torchvision	v0.9.0	v0.9.0	v0.9.0	v0.12
OpenCV - Python	v4.6.0.66	v4.6.0.66	v4.6.0.66	v4.7.0.72
Python	v3.6	v3.6	v3.6	v3.8

ACKNOWLEDGMENT

The authors would like to thank Niko Känsäkoski and Tapio Heikkilä from VTT for providing access and configuration support to Jetson AGX Xavier and Intel RealSense D415 devices.

REFERENCES

[1] F. Sherwani, M. M. Asad, and B. S. K. K. Ibrahim, "Collaborative robots and industrial revolution 4.0 (IR 4.0)," in *Proc. Int. Conf. Emerg. Trends Smart Technol. (ICETST)*, Mar. 2020, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9080724

[2] M. R. Endsley and W. Jones, "Situation awareness," in *The Oxford Handbook of Cognitive Engineering*, vol. 1, J. D. Lee and A. Kirlik, Ed. New York, NY, USA: Oxford Univ. Press, 2013, pp. 88–90.

[3] H. Levesque and G. Lakemayer, "Chapter 23: Cognitive robotics," *Found. Artif. Intell.*, vol. 3, pp. 869–886, Jan. 2008.

[4] VTT Technical Research Centre of Finland. (Aug. 16, 2021). *Press Release: VTT, Nokia & Sandvik Collaborate in 5G Powered Research Project on Next Generation Underground Mining Technology*. [Online]. Available: https://www.vttresearch.com/en/news-and-ideas/vtt-nokia-sandvik-collaborate-5g-powered-research-project-next-generation

[5] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[6] GitHub. (2023). *Ultralytics YOLOv5*. [Online]. Available: https://github.com/ultralytics/yolov5

[7] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei, and X. Wei, "YOLOv6: A single-stage object detection framework for industrial applications," 2022, *arXiv:2209.02976*.

[8] C.-Y. Wang, A. Bochkovskiy, and H.-Y. Mark Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," 2022, *arXiv:2207.02696*.

[9] D.-J. Shin and J.-J. Kim, "A deep learning framework performance evaluation to use YOLO in Nvidia Jetson platform," *Appl. Sci.*, vol. 12, no. 8, p. 3734, Apr. 2022.

[10] H. Gomes, N. Redinha, N. Lavado, and M. Mendes, "Counting people and bicycles in real time using YOLO on Jetson nano," *Energies*, vol. 15, no. 23, p. 8816, Nov. 2022.

[11] J. L. Chung, L. Ong, and M. Leow, "Comparative analysis of skeleton-based human pose estimation," *Future Internet.*, vol. 14, no. 380, pp. 11–23, Dec. 2022.

[12] A. A. Jeny, M. S. Junayed, and M. B. Islam, "PoseTED: A novel regression-based technique for recognizing multiple pose instances," in *Proc. 16th Int. Symp. Visual Comput.*, 2021, pp. 573–585, doi: 10.1007/978-3-030-90439-5_45.

[13] TensorFlow Hub. (2023). *Movenet Lightning*. [Online]. Available: https://tfhub.dev/google/movenet/multipose/lightning/1

[14] T. T. Than, D. K. D. Danh, H. L. Nguyen, and M. S. Nguyen, "Researching and implementing the posture recognition algorithm of the elderly on Jetson nano," in *Proc. Int. Conf. Multimedia Anal. Pattern Recognit. (MAPR)*, Oct. 2022, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9924968

[15] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, Dec. 2007.

[16] P. Pääkkönen and D. Pakkala, "Extending reference architecture of big data systems towards machine learning in edge computing environments," *J. Big Data*, vol. 7, no. 1, pp. 1–29, Apr. 2020.

[17] J. Fan, P. Zheng, and S. Li, "Vision-based holistic scene understanding towards proactive human–robot collaboration," *Robot. Comput.-Integr. Manuf.*, vol. 75, Jun. 2022, Art. no. 102304.

[18] GitHub. (2023). *Ultralytics YOLOv8*. [Online]. Available: https://ultralytics.com/yolov8

[19] W. F. Magalhaes, H. M. Gomes, L. B. Marinho, G. S. Aguiar, and P. Silveira, "Investigating mobile edge-cloud trade-offs of object detection with YOLO," in *Proc. Anais Do VII Symp. Knowl. Discovery, Mining Learn. (KDMiLe)*, Oct. 2019, pp. 49–56. [Online]. Available: https://sol.sbc.org.br/index.php/kdmile/article/view/8788

[20] Y. Matsubara, S. Baidya, D. Callegaro, M. Levorato, and S. Singh, "Distilled split deep neural networks for edge-assisted real-time systems," in *Proc. Workshop Hot Topics Video Analytics Intell. Edges*, Oct. 2019, pp. 21–26, doi: 10.1145/3349614.3356022.

[21] Y. Matsubara, D. Callegaro, S. Singh, M. Levorato, and F. Restuccia, "BottleFit: Learning compressed representations in deep neural networks for effective and efficient split computing," in *Proc. IEEE 23rd Int. Symp. World Wireless, Mobile Multimedia Netw. (WoWMoM)*, Jun. 2022, pp. 337–346. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9842809

[22] A. Malawade, M. Odema, S. Lajeuness-Degroot, and M. A. Al Faruque, "SAGE: A split-architecture methodology for efficient end-to-end autonomous vehicle control," *ACM Trans. Embedded Comput. Syst.*, vol. 20, no. 5, pp. 1–20, Sep. 2021.

[23] J. Lin, W. Chen, H. Cai, C. Gan, and S. Han, "Memory-efficient patch-based inference for tiny deep learning," in *Proc. 35th Conf. Neural Inf. Process. Syst.*, 2021, pp. 2346–2358. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/1371bccec2447b5aa6d96d2a540fb401-Paper.pdf

[24] TensorFlow. (2023). *Human Pose Classification with MoveNet and TensorFlow Lite*. [Online]. Available: https://www.tensorflow.org/lite/tutorials/pose_classification

[25] Y. Wang, M. Li, H. Cai, W. Chen, and S. Han, "Lite pose: Efficient architecture design for 2D human pose estimation," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, New Orleans, LA, USA, 2022, pp. 13126–13136. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022/papers/Wang_Lite_Pose_Efficient_Architecture_Design_for_2D_Human_Pose_Estimation_CVPR_2022_paper.pdf

[26] M. Yamazaki and M. E. Mori, "Rethinking deconvolution for 2D human pose estimation light yet accurate model for real-time edge computing," in *Proc. 16th IEEE Int. Conf. Autom. Face Human Gesture Recognit.*, Jodhpur, India, Dec. 2021, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9666963

[27] D. Maji, S. Nagori, M. Mathew, and D. Poddar, "YOLO-pose: Enhancing YOLO for multi person pose estimation using object keypoint similarity loss," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2022, pp. 2637–2646. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022W/ECV/papers/Maji_YOLO-Pose_Enhancing_YOLO_for_Multi_Person_Pose_Estimation_Using_Object_CVPRW_2022_paper.pdf

[28] H. Liu and L. Wang, "Collision-free human-robot collaboration based on context awareness," *Robot. Comput.-Integr. Manuf.*, vol. 67, Feb. 2021, Art. no. 101997.

[29] M. Verma, S. Kumawat, Y. Nakashima, and S. Raman, "YOGA-82: A new dataset for fine-grained classification of human poses," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 4472–4479. [Online]. Available: https://openaccess.thecvf.com/content_CVPRW_2020/papers/w70/Verma_Yoga-82_A_New_Dataset_for_Fine-Grained_Classification_of_Human_Poses_CVPRW_2020_paper.pdf

[30] Max Planck Institute Informatik. (2023). *MPII Human Pose Dataset*. [Online]. Available: http://human-pose.mpi-inf.mpg.de/

[31] I. Sittón-Candanedo, R. S. Alonso, J. M. Corchado, S. Rodríguez-González, and R. Casado-Vara, "A review of edge computing reference architectures and a new global edge proposal," *Future Gener. Comput. Syst.*, vol. 99, pp. 278–294, Oct. 2019.

[32] S. Martínez-Fernández, C. P. Ayala, X. Franch, and H. M. Marques, "Benefits and drawbacks of software reference architectures: A case study," *Inf. Softw. Technol.*, vol. 88, pp. 37–52, Aug. 2017.

[33] P. Pääkkönen, D. Pakkala, J. Kiljander, and R. Sarala, "Architecture for enabling edge inference via model transfer from cloud domain in a kubernetes environment," *Future Internet*, vol. 13, no. 1, p. 5, Dec. 2020.

[34] M. E. Arass, "Data life cycle: Towards a reference architecture," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 9, no. 4, pp. 5645–5653, Aug. 2020.

[35] M. Galster and P. Avgeriou, "Empirically-grounded reference architectures: A proposal," in *Proc. Joint ACM SIGSOFT Conf. Quality Softw. Architectures*, Boulder, CO, USA, 2011, pp. 153–158, doi: 10.1145/2000259.2000285.

[36] M. Farrajota. (2017). *Dbcollection*. [Online]. Available: https://dbcollection.readthedocs.io/en/latest/

[37] VideoLAN Organization. (2023). *VLC Media Player*. [Online]. Available: https://www.videolan.org/vlc/

[38] GitHub. (2023). *MMPose*. [Online]. Available: https://github.com/open-mmlab/mmpose

[39] *GOM & Company*. (2023). *GOM-Player*. [Online]. Available: https://www.gomlab.com/gomplayer-media-player/

[40] Roboflow Inc. (2018). *Roboflow*. [Online]. Available: https://roboflow.com/

[41] NVIDIA Corporation. (2023). *NVIDIA Tesla P100*. [Online]. Available: https://www.nvidia.com/en-us/data-center/tesla-p100/

[42] Roboflow Inc. (2022). *Roboflow Train: Understanding Training Graphs*. [Online]. Available: https://help.roboflow.com/faqs/roboflow-train-understanding-training-graphs

[43] Intel. (2018). *Introducing the Intel RealSense D400 Product Family*. [Online]. Available: https://www.intelrealsense.com/introducing-intel-realsense-d400-product-family/

[44] NVIDIA Corporation. (2023). *Jetson AGX Xavier*. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier/

[45] OpenCV Team. (2023). *OpenCV*. [Online]. Available: https://opencv.org/

[46] Python Software Foundation. (2023). *Python Multiprocessing*. [Online]. Available: https://docs.python.org/3/library/multiprocessing.html

[47] Eclipse Foundation. (2023). *Eclipse Mosquitto*. [Online]. Available: https://mosquitto.org/

[48] GitHub. (2023). *Jetson-Stats*. [Online]. Available: https://github.com/rbonghi/jetson_stats

[49] K. Wang, D. A. Rizqi, and H. Nguyen, "Skill transfer support model based on deep learning," *J. Intell. Manuf.*, vol. 32, pp. 1129–1146, Apr. 2021.

[50] P. Pääkkönen, J. Backman, D. Pakkala, J. Paananen, K. Seppänen, and K. Ahola, "Concept and architecture for applying continuous machine learning in multi-access routing at underground mining vehicles," *Appl. Sci.*, vol. 12, no. 20, p. 10679, Oct. 2022.

[51] NVIDIA Corporation. (2023). *Jetson Nano Developer Kit*. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[52] C. Wang, A. Bochkovskiy, and H. M. Liao, "Scaled-YOLOv4: Scaling cross stage partial network," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, Nashville, TN, USA, 2021, pp. 13029–13038. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/papers/Wang_Scaled-YOLOv4_Scaling_Cross_Stage_Partial_Network_CVPR_2021_paper.pdf
[53] C. Li, L. Li, Y. Geng, H. Jiang, M. Cheng, B. Zhang, Z. Ke, X. Xu, and X. Chu, "YOLOv6 v3.0: A full-scale reloading," 2023, *arXiv:2301.05586*.
[54] GitHub. (2023). *Torchdistill*. [Online]. Available: https://github.com/yoshitomo-matsubara/torchdistill
[55] X. Yuan, W. Ni, M. Ding, K. Wei, J. Li, and H. V. Poor, "Amplitude-varying perturbation for balancing privacy and utility in federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1884–1897, 2023.
[56] J. Lin, W. Chen, Y. Lin, J. Cohn, C. Gan, and S. Han, "MCUNet: Tiny deep learning on IoT devices," in *Proc. 34th Conf. Neural Inf. Process. Syst.*, 2020, pp. 11711–11722. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/86c51678350f656dcc7f490a43946ee5-Paper.pdf
[57] T. R. Gadekallu, G. Srivastava, M. Liyanage, M. Iyapparaja. C. L. Chowdhary, S. Koppu, and P. K. R. Maddikunta, "Hand gesture recognition based on a Harris hawks optimized convolution neural network," *Comput. Electr. Eng.*, vol. 100, May 2022, Art. no. 107836.
[58] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 4510–4520. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/papers/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.pdf

PEKKA PÄÄKKÖNEN received the double M.Sc. degree in information technology and in international business from the University of Oulu, Finland, in 2002 and 2015, respectively. He is currently a Senior Research Scientist with the VTT Technical Research Centre of Finland. His research interests include big data architectures, machine learning, databases, edge computing, stream processing, and service management.

DANIEL PAKKALA received the M.Sc. degree in electrical engineering and the Ph.D. degree in information processing science from the University of Oulu, Finland, in 2004 and 2020, respectively. He is currently a Principal Scientist and a Senior Project Manager (IPMA C) with the VTT Technical Research Centre of Finland. His research interest include machine intelligence, digital service architectures, edge computing, cognitive agents, and information management.