Container-Based IoT Architectures: Use Case for Visual Person Counting

Tiago Veiga[1,*], Hafiz Areeb Asad[2], Frank Alexander Kraemer[2] and Kerstin Bach[1,*]

[1]*Department of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway*
[1]*Department of Information Security and Communication Technology, Norwegian University of Science and Technology, Trondheim, Norway*

Abstract

The design and deployment of combined Internet-of-Things (IoT) and Artificial Intelligence (AI) solutions present several challenges, some of which are handled by modular logical models for the flow of data and containerized architectures that provide virtualized environments to encapsulate different functionalities in a network. The combinations of containers result in reusable and general solutions, even leading to visual composition solutions with recent deployment platforms. General requirements for these solutions are the capability of handling data acquisition, maintenance, and model training from zero information. This paper studies the deployment process for a use case of visual person counting from cameras located in outdoor areas. We show how a containerized solution fulfills the particular requirements for the use case, illustrating how the design of the modular architecture, data pipelines, and exposed services contribute to enhancing adaptive behavior through learning based on the context of the environment.

Keywords

AI Deployment, Container-based deployment, IoT architectures

1. Introduction

In a wide range of application domains, Internet-of-Things (IoT) and Artificial Intelligence (AI) are used in combination, such that IoT devices serve as a source of data which is collected and used by AI modules to give better insights and enable decisions about observed phenomena [1].

Deploying combined IoT/AI architectures brings several challenges [2], especially if we aim to provide general and reusable solutions. AI deployment platforms point towards this direction, recently shifting from focusing on unidirectional AI pipelines to more complex architectures. One example is the AI4EU platform [3], which combines the technical deployment platform with a community-maintained catalog of AI solutions and pipelines.

Cognitive architectures, coupling IoT and AI to drive autonomous and self-adaptive behavior of the sensor devices [4], lead to more complex behavior that optimizes system resources and application performance. In this context, networks with IoT devices can adapt to the context of the environment, both on the cloud and server sides. However, AI deployment platforms still lack support for features required by these combined architectures [5].

NAIS 2023: The 2023 symposium of the Norwegian AI Society, June 14-15, 2023, Bergen, Norway.
*Corresponding author.
tiago.veiga@ntnu.no (T. Veiga); hafiz.a.asad@ntnu.no (H. A. Asad); kraemer@ntnu.no (F. A. Kraemer); kerstin.bach@ntnu.no (K. Bach)

This paper studies the deployment process of a use case for person counting in outdoor areas from camera images. We describe the prerequisites that a deployment architecture must support for this case, which include reusability, learning, and forecasting functionalities. Then, we show our deployment solution, which follows a containerized standard and follows the guidelines for IoT/AI cognitive architectures. This case can be tied to other field applications, such as wildlife monitoring [6], person counting [7], or analysis of traffic density [8].

2. Use Case: Person Counting in Outdoor Areas

The application in our case study is based on person counting on outdoor areas to estimate the busyness in the observed areas. For the estimation, images from cameras deployed at several locations in Norway are publicly available in real time[1] and are updated at a fixed interval of 10 minutes, from which we choose a subset of locations around Trondheim. The learning process is intended to be fully autonomous and adapted to the local and changing environment, therefore, no data is available before deployment. From acquired data, statistics and predictions can either be used in a closed loop to improve the behavior of constrained-resourced devices or conveyed to external users such as people planning to go outdoors and who can use a prediction of movement to choose better when to go, or those responsible for the maintenance of tracks who can better plan their work.

Figure 1: Use case data pipeline: from data acquisition to forecast of busyness in the observed areas.

We focus on translating the logical data flow, summarized in Figure 1, to a solution that can be deployed. For that, we identify the following requirements that the solution should fulfill and will then describe an implementation that follows them.

- **R1** The solution should be reusable and flexible to changes in its components, allowing smooth updates of the global solution and transferability to similar scenarios.
- **R2** The architecture should allow learning spatial and/or temporal attention maps such that statistics for these trends can be provided based on historical data.
- **R3** The architecture should allow forecasting models to estimate future observations, such that it can support devices by planning their resources or providing information to external users.

3. Deployment Implementation

This section describes how we handle each requirement with a containerized solution.

[1]http://metnet.no/#kart_header (accessed on 30 May 2023)

3.1. R1: Reusability

Implementing an architecture for this use case follows a modular, containerized-based structure [5], that splits functionalities into different containers. This enhances reusability in two ways: first, it is possible to update a specific functionality without the need to change the whole architecture; second, it is compatible with recent AI deployment platforms, allows to reuse of publicly available containers from public catalogs, and provides a solution which can be transferred to other similar scenarios.

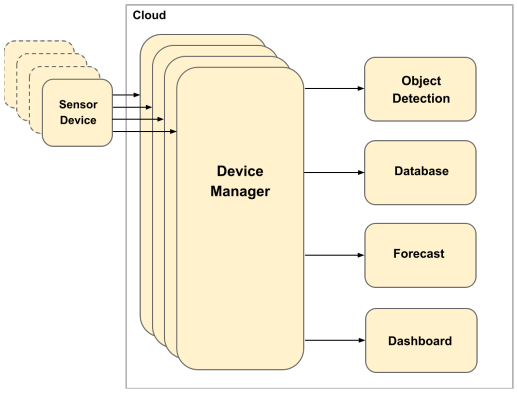

Figure 2: Module structure for the deployed solution. Sensor devices are not containerized as they typically are implemented as code running directly on devices. All the other modules are containers running in a cloud architecture. Arrows represent communication between the modules, in the form of microservices exposing data interfaces.

Figure 2 shows the container structure of the solution, illustrating sensor devices (not part of the containerized architecture) and container instances, which are all installed in a local server. To allow for more flexibility, each device's data is managed by a device manager container, acting as an orchestrator that connects external devices with modules in the network. Additional containers provide functionalities available to the managers: an object detection module that processes input images and detects bounding boxes of persons, which is reused from the catalog in the AI4EU platform; a database module that manages a MongoDB database and exposes services to access this information; a dashboard module that exposes an external web interface with information about the current status of the system; a forecast module that implements services to train prediction models and provide a forecast estimation from these models.

With this architecture, only data related to the number of detections is stored in the database, and images are discarded after being processed. Furthermore, the whole cloud architecture and, therefore, all data is circumscribed to the server where it is installed. In our case, this is a local server and therefore, data is passed through a local network.

3.2. R2: Learning Attention Maps

Resource-constrained IoT devices can select to drop the transmission of parts of an image as a mean to save energy, which is especially beneficial when areas that never contain any persons

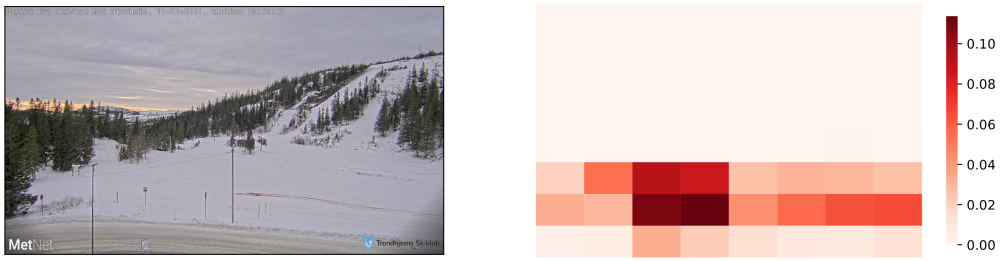

Figure 3: Visual attention model, as calculated by the device manager module over the entire range of images used as a test set. Darker cells are the image regions in which more often persons have appeared.

are in the field of view of a camera (e.g., in Figure 1 part of the image captures the sky). Each device manager container implements and maintains a visual attention map that can, by request, be sent to a device as the base for a tile selection policy.

The server receives the transmitted tiles, reassembles them with a background image received earlier, and then performs the object detection on these reduced tile sets. The details of the algorithms for the computation of the attention models and the policies are explained in [9].

Figure 3 shows an example of a spatial attention map for a camera, represented as a heatmap. Whenever persons are detected in a newly received image, the heatmap is increased in the respective image slices where bounding boxes are located. Cells in the heatmap with darker coloring correspond to areas where the detection of persons is more likely. Similar attention maps can be made for temporal maps, showing the periods during each day with more detections.

3.3. R3: Forecasting

This architecture includes a forecast module, which implements the functionalities for the forecasting procedure. It exposes services for initialization, model training, and prediction. At initialization, it receives configuration data such as the number of weeks included in the training data. Then, whenever the manager calls for model training or requests a new prediction, the forecast module receives the current timestamp, constructs the needed datasets with calls to the database, and updates or runs predictions, respectively, with the stored trained model.

3.3.1. Data

To evaluate the method, we collected a dataset of detections for eight cameras between December 2021 and January 2023, some of which are co-located but with non-overlapping fields of view, using the *You only look once approach* (YOLO) [10] model as the detection algorithm. The output is filtered to store only persons detected with a confidence threshold higher than 0.5, above the default 0.25 value used in YOLO, to reduce the number of false positives. Additionally, meteorological data is obtained through a public API from the Norwegian Meteorological Institute [2]. Figure 4 illustrates person counts for two sample periods, during winter and summer, along with precipitation data. These are good examples of how weather influences the presence of people in outdoor areas, such that fewer people are counted when there is more precipitation.

[2]https://frost.met.no/index.html (accessed on 30 May 2023)

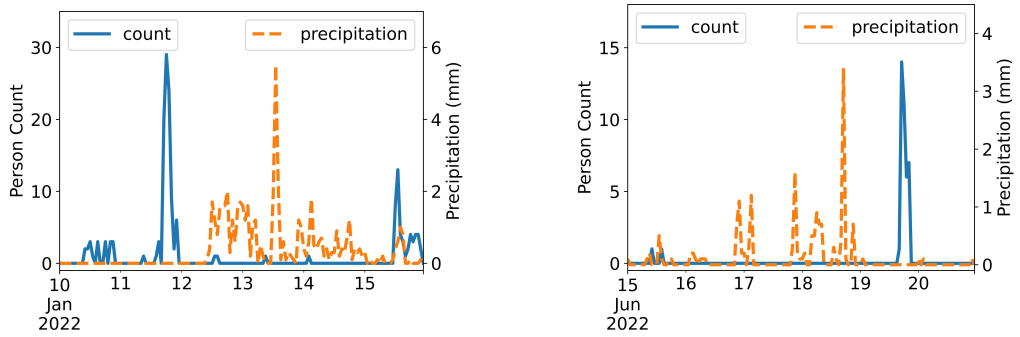

Figure 4: Sample of person count and precipitation data for camera *jervskogen1* for winter and summer periods.

3.3.2. Training Procedure

The procedure for training the forecast model follows a rolling window fashion, which is visually illustrated in Figure 5. In this procedure, the forecast model is retrained weekly (specifically, we selected training on Sundays at midnight). At this training point, data from the previous week is collected and used for training a model that forecasts the next 24 hours. Then, a decision step happens daily (every midnight) in the following week, in which the trained model is used to forecast the next day. The procedure repeats weekly with a new training step, for which we discard old data and incorporate the observations from the past week. The output of the training step is a selection of the best model using a randomized search on hyperparameters.

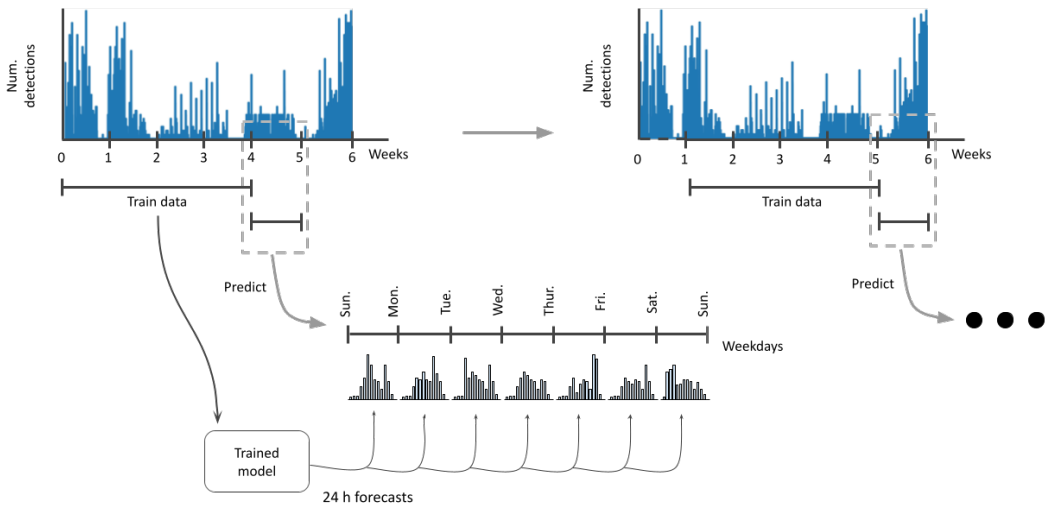

Figure 5: Illustration of the data split and train procedure. At each training point, data from the previous four weeks is used as training data for the forecast model, which, in turn, is used every midnight to predict the next 24 hours. After one week, the model is retrained with an updated dataset (discards older data and incorporates the observations from the past week), and the procedure repeats.

3.3.3. Results

To evaluate this approach, we compare it with real measured detections using the root mean squared error (RMSE). Additionally, we compare those results with a baseline which, for each hour, computes the expected detections as the average of detections at the same hour and weekday in the past four weeks, with a summary in Table 1. Despite being a difficult forecasting problem, due to the imbalance in the dataset towards many zero detections and the 10 minute interval between images, the error reduces when external features are included in the model.

	RMSE	
Camera	Our	Statistical Average
jervskogen1	**1.43 ± 1.46**	1.60 ± 1.45
jervskogen2	**1.11 ± 2.31**	1.23 ± 2.31
nilsbyen2	**1.67 ± 2.34**	1.82 ± 2.27
nilsbyen3	**3.41 ± 4.02**	3.8 ± 3.92
skistua	**1.58 ± 2.07**	1.64 ± 1.92
ronningen1	**2.70 ± 2.10**	2.78 ± 2.09
ronningen2	**1.92 ± 1.75**	2.16 ± 1.77
meraker1	**0.92 ± 2.14**	1.2 ± 2.47

Table 1
RMSE between real observed detections with: **Our)** forecast using the trained model with our procedure; **Statistical Average)** forecast in which predictions are the average of the previous weeks, for the same weekday and hour. **Train features**: Weekday, hour, weather history, weather forecast, detection history (average for the last 4 weeks for each hour of the day)

4. Conclusions

We present a deployment solution for a use case of visual person counting from cameras in outdoor areas, combining IoT devices and AI components. The implemented solution is a reusable containerized architecture, allowing the inclusion of more devices after deployment. Using a manager component allows centralized management of the data flow and keeps updated attention maps. The separation of different functionalities in each container allows to add services to the system, including a service to forecast observed phenomena.

This implementation can contribute to the efficient operation of resource-constrained devices. Our use case includes energy management for devices with a limited energy budget, which can benefit from predictions on the busyness of the observed area for a more efficient use of energy towards periods when more persons are expected.

Acknowledgments

This work has been partly funded by the SFI NorwAI, (Centre for Research-based Innovation, 309834) and by the European Union's Horizon 2020 research and innovation project AI4EU, grant agreement No 825619.

References

[1] J. Zhang, D. Tao, Empowering things with intelligence: A survey of the progress, challenges, and opportunities in artificial intelligence of things, IEEE Internet of Things Journal 8 (2021) 7789–7817. doi:10.1109/jiot.2020.3039359.

[2] A. Paleyes, R.-G. Urma, N. D. Lawrence, Challenges in deploying machine learning: A survey of case studies, ACM Comput. Surv. (2022). doi:10.1145/3533378.

[3] P. Schüller, J. P. Costeira, J. Crowley, J. Grosinger, F. Ingrand, U. Köckemann, A. Saffiotti, M. Welss, Composing complex and hybrid ai solutions, 2022. doi:10.48550/ARXIV.2202.12566.

[4] A. E. Braten, F. A. Kraemer, D. Palma, Autonomous iot device management systems: Structured review and generalized cognitive model, IEEE Internet of Things Journal 8 (2021) 4275–4290. doi:10.1109/JIOT.2020.3035389.

[5] T. Veiga, H. A. Asad, F. A. Kraemer, K. Bach, Towards containerized, reuse-oriented ai deployment platforms for cognitive iot applications, Future Generation Computer Systems 142 (2023) 4–13. doi:https://doi.org/10.1016/j.future.2022.12.029.

[6] A. R. Elias, N. Golubovic, C. Krintz, R. Wolski, Where's the bear? automating wildlife image processing using iot and edge cloud systems, in: Proceedings of the Second International Conference on Internet-of-Things Design and Implementation, IoTDI '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 247–258. doi:10.1145/3054977.3054986.

[7] A. B. Chan, Z.-S. J. Liang, N. Vasconcelos, Privacy preserving crowd monitoring: Counting people without people models or tracking, in: 2008 IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–7. doi:10.1109/CVPR.2008.4587569.

[8] L. Ciampi., C. Santiago., J. P. Costeira., C. Gennaro., G. Amato., Domain adaptation for traffic density estimation, in: Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP„ INSTICC, SciTePress, 2021, pp. 185–195. doi:10.5220/0010303401850195.

[9] H. A. Asad, F. A. Kraemer, K. Bach, C. Renner, T. S. Veiga, Learning attention models for resource-constrained, self-adaptive visual sensing applications, in: Proceedings of the Conference on Research in Adaptive and Convergent Systems, RACS '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 165–171. doi:10.1145/3538641.3561505.

[10] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, TaoXie, J. Fang, imyhxy, K. Michael, Lorna, A. V, D. Montes, J. Nadar, Laughing, tkianai, yxNONG, P. Skalski, Z. Wang, A. Hogan, C. Fati, L. Mammana, AlexWang1900, D. Patel, D. Yiwei, F. You, J. Hajek, L. Diaconu, M. T. Minh, ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference, 2022. doi:10.5281/zenodo.6222936.