Object detection in driving datasets using a high-performance computing platform: A benchmark study

TAHIR EMRE KALAYCI, GABRIELA OZEGOVIC, BOR BRICELJ, MARKO LAH, ALEXANDER STOCKER

Virtual Vehicle Research GmbH, Inffeldgasse 21A, 8010 Graz, Austria

ABSTRACT

Nowadays, machine learning methods are increasingly used in different parts of autonomous driving and driving assistance systems. Yet, data and computational requirements can be enormous with these methods. Thus, providing several datasets containing many and diverse cases for the target problem and sufficient hardware for training and application of ML methods are too critical for achieving accurate results when applying them. Hence, we present an object detection benchmark study implementing the knowledge graph-based data integration framework to meet the data requirements and run the implementation on a big data and high-performance computing (HPC) platform, namely the EVOLVE. We applied different object detection methods to widely known open datasets, and compared the results on three different hardware setups, including EVOLVE. We also performed a small-scale transfer learning experiment. The results show that EVOLVE allowed the exploitation of much bigger data leading to a more efficient application of the object detection models with the help of the knowledge graph-based data integration framework. EVOLVE significantly improved the execution times compared to running them on a local laptop and a virtual machine and provided the easy-to-use and ready-to-use means to store large datasets and apply different models with its hardware and software stack.

INDEX TERMS

Big Data, Driving Datasets, EVOLVE, High Performance Computing, Object Detection, nuScenes, Transfer Learning

I. INTRODUCTION

Today, there is increasing usage of autonomous driving and driving assistance systems, and they require and depend on safe and reliable software systems. Although these software systems consist of different parts, various machine learning (ML) methods are used to provide detection, prediction, and decision support. Machine learning, together with data analytics, can help learning underlying patterns and make decisions based on the insights obtained [1]. Therefore, our confidence in trusting predictions of these ML methods and estimating this confidence is one of the most relevant concerns [2]. Hence, in many real-world cases, generalization, providing accurate predictions on unseen data depending on the characteristics of the target problem, is very important. Generalization enables the usage of the method in diverse situations to solve the target problem. However, ML methods are usually prone to issues and missing scenarios in the training data, which may hinder the generalization. One suggested solution is to provide several datasets containing many diverse cases for the target problem during the training. Hence, it is essential to find and utilize datasets containing various scenarios for achieving a functional, efficient and trustworthy ML method.

Consequently, data integration [3], the process of combining different data that are stored in a set of separate data sources into a single unified view, is very important for accessing and using different scenarios for various purposes [4]. Integration should provide a way of standardizing data, improving accuracy and productivity, and greater flexibility and agility [5]. Additionally, roles in data science (DS) and machine learning (ML) teams, who are working in different stages of the DS/ML life cycle, are very diverse [6]. However, there are not enough DS/ML professionals to fill the job demands, and 80% of their time is generally spent on...
Vision-based object detection is an important task for autonomous driving [13]. It is important to solve this problem over the images for safe and efficient autonomous driving and support the driver in many risky situations. Vision-based object detection has the challenges of detecting objects in images, estimating their position, and finally estimating their classes [14]. One common approach is training object detectors that operate on a sub-image and exhaustively applying these detectors across all locations and scales [15]. Recently, various methods based on deep learning, especially Convolutional Neural Network (CNN), have been successfully proposed to overcome these challenges. Today, a modern detector is usually composed of two parts, a backbone, which is a pre-trained model, and a head, which is used to predict classes and bounding boxes of objects [16]. There are different backbones for GPU (VGG [17], ResNet [18], ResNeXT [19], DenseNet [20], etc.) and CPU (SqueezeNet [21], MobileNet [22], ShuffleNet [23], etc.) training [16]. The head part is usually categorized as a one-stage object detector or a two-stage object detector. First, we have chosen YOLOv4 [16], which is a one-stage object detector or a two-stage object detector.

We aim to have a modern and diverse set of models for comparison by these four models.

In the last years, convergence between high-performance computing (HPC) and big data analytics (BDA) has become an established research area that provided new opportunities for unifying the platform layer and data abstractions in this ecosystem [32]. Investigating the benefits of using a big data and high-performance computing (HPC) platform, specifically the EVOLVE [33], [34], for different use cases is essential. EVOLVE is a pan-European Innovation Action building a converged infrastructure to bring the HPC, cloud, and big data worlds together. EVOLVE’s advanced computing platform combines HPC-enabled capabilities with transparent deployment at a high abstraction level, and a versatile Big-Data processing stack for end-to-end workflows. EVOLVE’s hardware and software stack supports large-scale, data-intensive applications, driven primarily by industry requirements set by pilot and proof-of-concept use cases from diverse fields. Given the unprecedented data growth we are experiencing, EVOLVE’s infrastructure is essential in enabling the cost-effective processing of massive amounts of data and the adoption of multiple high-end technologies. The platform addresses the processing of massive data that requires demanding computation by offering a novel, integrated computing environment. The intention is to boost productivity and allow interoperability while maintaining hardware-specific performance benefits.

Our objective is to demonstrate an object detection use case using the knowledge graph-based data integration framework, which utilizes the high-performance computing platform. We have selected nuScenes dataset provided by the Motional and Level 5 Perception Dataset provided by Lyft for the demonstration. We implement the previously proposed knowledge graph-based data integration [9] to combine these two different public datasets to achieve a successful data integration for the easy application of object detection methods over EVOLVE platform. We apply the object detection task using YOLOv4 and three additional models provided from TensorFlow Hub [25] over three different hardware setups. We summarize our contributions as the following:

- We introduce the EVOLVE big data and high-performance computing platform,
- We demonstrate the application of the knowledge-graph based data integration framework,
- We present the results of applying different object detection methods to widely known open datasets,
- We compare three different hardware setups to show the benefits of EVOLVE,
- We also present a small-scale transfer learning experiment using an object detection model to detect traffic

References:
- **CenterNet** Object detection model [26] with the Hourglass backbone [27] training images scaled to 512x512
- **EfficientDet** object detection model (SSD with EfficientNet-b0 + BiFPN feature extractor, shared box predictor and focal loss) [28], [29], training images scaled to 512x512
- **Faster R-CNN** [30] with Resnet-50 V1 [18] object detection model, with training images scaled to 640x640

Links:
- https://tfhub.dev/tensorflow/centernet/hourglass_512x512/1
- https://tfhub.dev/tensorflow/efficientdet/d0/1
- https://tfhub.dev/tensorflow/faster_rcnn/resnet50_v1_640x640/1
lights.

The remainder of this paper is organized as follows: we present the EVOLVE platform in Section II, we describe the datasets in Section III and we summarize the methodology in Section IV. We show the results in Section V, and finally, we conclude the paper in Section VI.

II. BIG DATA AND HIGH-PERFORMANCE COMPUTING PLATFORM EVOLVE

Applying Machine Learning methods is a growing trend in autonomous driving and assistance systems. While current systems produce immense volumes of data, at the same time, ML methods require a lot of diverse data and a complete iterative lifecycle with a human-in-the-loop perspective for a good performance. Also, ML methods usually require plenty of computing resources for training and inference tasks. High-performance computing platforms can help to lessen this burden.

Because such systems require and depend on safe and reliable methods, monitoring and understanding the applied techniques and their results are critical for safety-critical applications with humans involved. Therefore, humans need to understand and anticipate the actions taken by the autonomous systems for trustful and safe cooperation [35]. In response to this problem, the concept of explainable AI (XAI) was introduced, which refers to machine learning techniques that provide details and reasons that make a model’s mechanism easy to understand [35], [36].

With these in mind, we aim to investigate the benefits of using HPC with high-performance big data storage and processing for object detection use case that relies on a knowledge graph-based data integration framework [9]. We will utilize the already trained YOLOv4 [16], CenterNet HourGlass104 512x512, EfficientDet D0 512x512, and Faster R-CNN ResNet50 V1 640x640 object detection models for the use case. We explain these models in Section IV-A.

EVOLVE [34] is an infrastructure aiming to bring the HPC, cloud, and big data worlds together. It supports large-scale, data-intensive applications driven primarily by industry requirements set by pilot and proof-of-concept use cases from diverse fields. The applications express their logic and required datasets in the form of workflows that can be automated, shared, refined, and maintained across groups of domain experts without significant IT expertise. It is possible to manipulate all workflows and data through a web-based dashboard. The code (expressed in portable notebook formats) is executed seamlessly on HPC hardware, using a rich and versatile set of big data processing frameworks.

The EVOLVE testbed is shared across applications using a Kubernetes-based execution framework. All workflow stages are containerized, facilitating ease of deployment, isolation, portability, and reproducibility. At the same time, the testbed includes provisions for data protection (security and integrity), the main reason that hinders shared deployment in domains with sensitive data.

Its unique features are the following [34]:

- complex application orchestration, it is fully containerized and controlled through a workflow definition language,
- A unified data management layer that allows users to transparently access their local or Cloud-based datasets as files, both through the dashboard and in containers,
- Custom, fine-grain resource allocation, and scheduling schemes, as well as accelerator sharing methods and policies,
- Tight accelerator integration with big data frameworks, such as Spark, both for batch processing and streaming applications,
- Seamless execution of HPC applications as workflow stages, with Slurm Workload Manager [37] compatibility,
- Embedded visualization modules with innovative and responsive features, allowing efficient and versatile interaction with data,
- Efficient monitoring of utilization, performance, and QoS for the whole system and individual workflows.

The integrated technologies of EVOLVE’s testbed, ranging from the hardware infrastructure to system-level and software-level deployments are summarized in Figure 1 [33].

The EVOLVE combines 16 heterogeneous Intel x86 compute nodes (Broadwell, Haswell, and Skylake families), with various accelerators and storage technologies installed, as described in Table 1 [33]. Accelerators (GPUs and FPGAs) are installed in six nodes, while all nodes are interconnected via NVIDIA Mellanox InfiniBand FDR links (56 Gb/s) and run GNU/Linux. The EVOLVE hardware platform adopts the Intel Xeon CPUs, NVIDIA Tesla K20, P40, and V100 GPUs supporting several programming models (like CUDA, OpenMP, and OpenCL), as well as accelerator sharing methods and policies.

**TABLE 1: Advanced Computing Platform** [33]

| CPUs     | Intel Xeon [Platinum 8153, E5-2690/2670/2470] |
|----------|-----------------------------------------------|
| GPUs     | Nvidia [Tesla V100, P40, K20Xm]               |
| FPGAs    | Altera Arria 10, Stratix 10                   |
| Storage  | IME, Lustre, NFS                              |

FIGURE 1: Overview of EVOLVE platform [33]
OpenMP, OpenAcc, and OpenCL), and Altera Arria 10, Stratix 10 FPGAs, used via native HDL (VHDL, Verilog) and Intel OpenCL SDK. Network File System serves the home directories for convenience. For high-performance I/O operations, 120TB Lustre open-source and parallel file system [38] implemented with 2 NetApp FAS2700 Series storage arrays are integrated with DDN’s Infinite Memory Engine (IME) [39], a scale-out, software-defined, flash storage platform that automatically optimizes and coordinates data movement from high-speed devices to the high-volume back end. Hence, the path from compute nodes to storage devices is accelerated [33, 34].

The heterogeneous Hardware infrastructure is integrated into a single execution environment using Kubernetes [40], [41] with The SkyNet [42], Volcano [43], and the interference-aware custom scheduler [44] extensions. For the unified storage layer EVOLVE uses open-source software frameworks such as Karvdash [45], Datashim [46], and H3 [47].

The software frameworks, run-time components for workflow execution are packaged up in containers as microservices. These containers are used as building blocks for workflow steps. The containerized executor instances of Apache Kafka [48], Apache Spark [49], TensorFlow [24], MPI Forum [50], Dask [51], etc. preserve the user isolation/property rights of the execution framework (i.e., run in separate name-spaces), use the custom Kubernetes resource management extensions where needed, and send performance metrics to monitoring. More importantly, microservices exploit the fast storage, accelerators, and high-speed low-latency network capabilities of the hardware infrastructure.

III. DATASETS

In this section we present the nuScenes data format we are using (Section III-A), datasets (Section III-B and Section III-C), and process of importing them to a knowledge graph (Section III-D).

A. nuScenes DATA FORMAT

In this study, we use two public datasets. The common feature of these datasets is the nuScenes data format [52]. nuScenes data format is easy to read, import, and manage. It is stored in a lightweight, text-based, and language-independent syntax JSON [53]. All annotations and metadata (including calibration, maps, vehicle coordinates, etc.) are covered in relational tables (See fig. 1). Every row has a unique primary key token for the identification. Foreign keys are used to link to other tables, for example sample_token refers to the token of the table sample.

B. MOTIONAL. nuScenes DATASET

The nuScenes dataset is a publically available large-scale dataset for autonomous driving developed by the team at Motional [54]. It is collected from a fully autonomous vehicle sensor suite: 6 cameras, five radars, and one LIDAR, all with a full 360-degree field of view. It comprises 1000 scenes, each 20s long and fully annotated with 3D bounding boxes for 23 classes and eight attributes. It has seven times as many annotations and 100 times as many images as the pioneering KITTI dataset. It is under CC-BY-NC-SA-4.0 license [55], which means that anyone can use it for non-commercial research purposes. All data, code, and information are made available online.

C. LYFT LEVEL 5 PERCEPTION DATASET

Lyft Level 5 Perception Dataset is also a public large-scale dataset using the same nuScenes data format [56]. This dataset features the raw LIDAR and camera inputs collected by autonomous vehicles within a bounded geographic area. These autonomous vehicles contain an in-house sensor suite that collects raw sensor data on other cars, pedestrians, traffic lights, etc. The dataset includes 1.3 million 3D annotations, 30000 LIDAR point clouds, and more than 350 scenes 60-90 minutes long. It is under the CC-BY-NC-SA-4.0 license. They have their own modified development kit.

D. IMPORTING THE DATA TO THE KNOWLEDGE GRAPH

One necessary step is importing the datasets to the knowledge graph. Neo4j graph database [57] stores the knowledge graph. The main reason to store data as a knowledge graph is to combine different data that are stored in a set of separate data sources into a single, unified view [4], [9].

https://github.com/nutonomy/nuscenes-devkit
https://github.com/lyft/nuscenes-devkit

FIGURE 2: Schema of the nuScenes data format [52]
Hence, the knowledge graph-based data integration provides a flexible, standardized, and agile platform to access and manipulate the datasets through a unified, single access layer. The framework provides a simple to use library to access different adapters and integrated data that is available in the graph database and a JupyterHub Notebooks component, which utilizes Jupyter notebooks for storing, implementing, and executing adapters and algorithms, which improves computational reproducibility by simplifying code reuse. Therefore, we prepared our nuScenes adapter and algorithms as Jupyter notebooks in an iterative process. The library provides functionality for creating adapter notebooks for different data formats on-demand and also functions to present static notebooks to the user, which are prepared specifically for the Lyft and Motional datasets.

These nuScenes adapter notebooks implement a step-by-step data import process by using variables from the user and knowledge graph connection information (See Figure 3). Another benefit of using the framework is the ability to execute importing process asynchronously using Apache Celery and RabbitMQ. Apache Celery is a distributed task queue to distribute work across threads or machines. Clients communicate with workers through messages, and different workers perform their tasks asynchronously. RabbitMQ is open-source message-broker software that provides a communication structure between different services using the Advanced Message Queuing Protocol (AMQP). It is used as a message broker, to pass the messages between the framework library and Apache Celery workers. Therefore, the framework library sends the import queries to the Celery worker on the platform through a message on the RabbitMQ queue. When the worker reads the message, it will start executing the import queries on the Neo4j. These queries read the JSON files containing data and import these data to the graph by creating nodes and relations.

To import Motional and Lyft datasets, we implemented a specific adapter supporting nuScenes data format. This adapter processes the raw data and imports the necessary parts representing the nodes and relations. But it keeps the binary files such as images and LIDAR output in their respective folder to access them when needed.

After this import process, a database is available in the graph database in the nuScenes schema (Figure 4), which is just the same as the schema of the original data format. Now it is ready to use with Cypher query language, which is a well-established declarative language for querying and updating property graph databases.

After the import process, we have a different number of nodes and relationships in different environments. The main reason is the usage of different versions of each dataset in different environments. While we imported a mini version of Motional and a sample version of Lyft dataset in the development laptop and a mini version of Motional and a complete training dataset of Lyft in the virtual machine for performance issues, we imported full training versions of both datasets in the EVOLVE for the experiment. Therefore, you see a different number of nodes and relations for the datasets in the resulting databases of each environment Table 2. Through the EVOLVE platform, we are able to create 6,538,057 nodes in 2101.52 seconds for Motional, and 1,047,118 nodes in 445.6 seconds for Lyft, respectively. Although pre-EVOLVE structures deliver 82,454 nodes in 22.7 seconds and 9,442 nodes in 4.14 seconds, EVOLVE allowed the exploitation of much larger knowledge graphs leading to more efficient object detection models. For instance, we didn’t have enough space on the Virtual Machine to load the entire Motional dataset.

IV. METHODOLOGY
For the model execution, we prepared separate Jupyter notebooks implementing different steps to apply each model.
TABLE 2: Nodes and relationships resulting from data import. We imported only sample versions of datasets in local laptop, while imported complete training dataset of Lyft in Virtual Machine and sample version of Motional dataset.

| Environment       | Dataset  | Nodes  | Relations | Node Import Time | Relationship Import Time |
|-------------------|----------|--------|-----------|------------------|--------------------------|
| EVOLVE            | Motional | 6538057| 21650288  | 2101.52s         | 3954.91s                 |
|                   | Lyft     | 1047118| 3677257   | 445.60s          | 1545.94s                 |
| Virtual Machine   | Motional | 82454  | 214500    | 6.68s            | 18.82s                   |
|                   | Lyft     | 1047118| 3677257   | 94.81s           | 319.33s                  |
| Local Laptop      | Motional | 82454  | 214500    | 22.7s            | 27.88s                   |
|                   | Lyft     | 9442   | 37975     | 4.14s            | 8.84s                    |

FIGURE 5: Steps for the application of the object detection models

to the images available in each dataset Figure 5. We executed each notebook for each environment separately. The notebook first retrieves the sample_data nodes from the knowledge graph constructed in the import phase using the framework’s library. Then, the notebook reads the image file from the dataset folder for each node. Finally, it performs object detection over the image using the corresponding model. At the end of execution, detection results are stored in the knowledge graph (Section IV-B).

A. OBJECT DETECTION MODELS

We implemented notebooks for the following object detection models:

- We implemented a YOLOv4 object detection model [16], which is an efficient and powerful object detection model. It is a modification of the state-of-the-art methods (such as CBN [63], PAN [64], CBAM [65]), which tries to make them more efficient and suitable for single GPU training. It is available as an Open Source repository through YOLO license.

  https://github.com/AlexeyAB/darknet

- **CenterNet HourGlass104 512x512** is CenterNet object detection model [26] with the Hourglass backbone [27]. It is trained on COCO 2017 dataset [31] with training images scaled to 512x512. CenterNet [26] represents objects by a single point at their bounding box center, and other properties, such as object size, dimension, 3D extent, orientation, and pose, are then regressed directly from image features at the center location. It is end-to-end differentiable, simpler, faster, and more accurate than corresponding bounding box-based detectors and achieves the best speed-accuracy trade-off on the COCO dataset [26]. Hourglass backbone [27] is a novel convolutional neural network architecture, processing and consolidating features across all scales to capture the various spatial relationships. It uses repeated bottom-up, top-down processing used in conjunction with intermediate supervision to improve the performance of the network [27].

- **EfficientDet D0 512x512** is EfficientDet Object detection model (SSD with EfficientDet-b0 + BiFPN feature extractor, shared box predictor and focal loss) [28], [29]. It is trained on COCO 2017 dataset [31] with training images scaled to 512x512. Tan et al. [28] developed EfficientDet family of object detectors by proposing a weighted bi-directional feature pyramid network (BiFPN). This network allows easy and fast multi-scale feature fusion and a compound scaling method that uniformly scales the resolution, depth, and width for all backbone, feature network, and box/class prediction networks at the same time [28]. Their scaled EfficientDet achieves state-of-the-art accuracy with much fewer parameters and FLOPs than previous object detection and semantic segmentation models.

- **Faster R-CNN ResNet50 V1 640x640** is Faster R-CNN [30] with ResNet-50 V1 [18] Object detection model. It is trained on COCO 2017 dataset [31] with training images scaled to 640x640. Faster R-CNN [30] introduces a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. It is trained end-to-end to generate high-quality region proposals. These region proposals are then used by Fast R-CNN for detection [30]. ResNet-50 V1 [18] improves the training of substantially...
deeper networks by employing a residual learning framework. It reformulates the layers as learning residual functions regarding the layer inputs [18].

B. STORING THE RESULTS AND REPRODUCIBILITY

All the detection results for each image are stored in the knowledge graph as nodes (Figure 6). These detection nodes are also linked to the sample_data of the image with relationships. Detection nodes contain detected class name and predefined properties (algorithm UUID, AXE, role, timestamp, etc.). These predefined properties allow us to track different runs and the provenance of the node. Relationships contain the bounding box coordinates, confidence and same predefined properties (See Figure 7 for an example result with two-car detections, only a subset of properties are listed). Hence, it is possible to compare the results of different algorithm executions over the same image using these stored properties.

FIGURE 6: We create a separate node for each of the class available in the result, here you can see all classes with some selected sample_data nodes

FIGURE 7: Two cars are detected on this sample data, we only list prediction related property values and timestamp of the run

It is also possible to store the detection results over images after applying the models. That helps to compare the models easily. For example, when we apply these models to the same image, we can easily see and analyze the differences between detection results of each model:

- YOLOv4 detects 7 cars and 4 traffic lights (Figure 8a)
- CenterNet with HourGlass104 backbone detects 13 cars, 6 traffic lights and 1 truck (Figure 8b)
- EfficientDet D0 detects 6 cars and 2 traffic lights (Figure 8c)
- Faster R-CNN ResNet50 V1 detects 8 cars, 3 traffic

FIGURE 8: Detection results over the same image with different models
TABLE 3: Comparison of running detection models on different platforms over the images from the KG (Italic values are the number of images that are completed in 1h execution)

| Image Count | Local Laptop | Virtual Machine | EVOLVE HPC |
|-------------|--------------|-----------------|------------|
| 100         | 26.05        | 34.93           | 1.25       |
|             | 224          | 167             | 8.98       |
| (a) YOLOv4  |              |                 |            |
| 1000        | 68.77        | 88.41           | 2.05       |
|             | 85           | 68              | 16.25      |
| (b) CenterNet HourGlass104 |              |                 |            |
| 100         | 44.80        | 62.51           | 1.87       |
|             | 135          | 94              | 11.42      |
| (c) EfficientDet D0 |              |                 |            |
| 1000        | 23.02        | 29.63           | 1.35       |
|             | 263          | 229             | 8.58       |
| (d) Faster R-CNN with Resnet-50 V1 |              |                 |            |

lights and also wrongly detects 2 people (Figure 8d).

V. EXPERIMENTS AND RESULTS

We present the results of two experiments performed in this study.

A. APPLICATION OF OBJECT DETECTION MODELS

We have used the following four different object detection model in the experiments as explained in previous section (Section IV-A): a) YOLOv4, b) CenterNet HourGlass104 512x512, c) EfficientDet D0 512x512, and d) Faster R-CNN ResNet50 V1 640x640. We have run each object detection model in the following different environments:

- **Local Laptop** is the developer laptop, where we have run experiments on a JupyterLab deployed as a Docker container.
- **Virtual Machine** is an Ubuntu LTS GNU/Linux operating system installed Virtual Machine with 64GB RAM and 8 core Intel(R) Xeon(R) Gold 6146 CPU @ 3.20GHz. We have run all experiments on a JupyterHub deployed as a Docker container.
- **EVOLVE** is the HPC platform (Section II), where we have run the experiments using Argo Workflows in order to achieve seamless parallel execution in the context of Kubernetes. Experiments are parallelized using 100 nodes. An essential benefit of EVOLVE platform is the high-performance big-data storage and processing with HPC features, including acceleration, large amount of memories, fast storage architectures, and high-speed interconnections.

The comparison of running detection models on these different platforms is listed in Table 3. We have run each detection model for 100 images and 1000 images separately.

Results for 100 images are shown in Figure 9. EVOLVE’s infrastructure provides up to 30x speedup for the YOLOv4, 34x speedup for the CenterNet HourGlass104, 24x speedup for EfficientDet D0, and 17x speedup for Faster R-CNN with ResNet-50 v1 for 100 images compared to the Local Laptop. For 1000 images we have reached execution time limit (1h) in Local Laptop and Virtual Machine. Hence, comparing the results using a chart is not meaningful. However, EVOLVE platform provided the means to complete object detection on all these images in tested models in less than 20 minutes.

B. APPLICATION OF TRANSFER LEARNING

We performed a small-scale transfer learning experiment by retraining the EfficientDet D0 512x512 object detection model from TensorFlow Model Hub to detect traffic lights in images (Figure 10). We have chosen a smaller model architecture to retrain because of the limitations of local computational resources and time pressure of the ongoing project related to the EVOLVE. The results of this transfer learning experiment are as follows:

- **Using a synthetic dataset**: we used 123 images from Udacity’s CARLA Test Site Simulator, which have accurate bounding boxes with low variability in observed scenarios and lighting conditions. We ended up with lower confidence in detections and more false positives.
- **Using a real-world dataset**: we used 3078 images from Bosch Small Traffic Lights, which have potential problems in bounding boxes, such as size, occlusion, coordinates mismatch, but offer a higher variability of scenarios with similar lighting conditions. We ended up with higher confidence in detections and better localized and classified objects.

First, we attempted to perform the transfer learning experiment in the Google Colab environment with a free subscription and then on EVOLVE platform. We chose the Google Colab because running the experiment on local resources was computationally insufficient, even if we preferred a small model architecture and tuned down hyperparameters (batch size and image resolution). But, the Google Colab environment with a free subscription was also
insufficient due to the frequent crashes and long training times; it took 6 hours for the baseline demo. Contrary to this, retraining with EVOLVE platform took only 1.5 hours, which was a significant advantage over the Google Colab session. We utilized the GPU resources on EVOLVE, which improved the results significantly.

VI. CONCLUSIONS

We demonstrated an object detection benchmark study using the knowledge graph-based data integration framework. We presented the results of applying the framework to widely known open datasets to show the benefits of using EVOLVE platform. We have used the nuScenes dataset provided by the Motional and Level 5 Perception Dataset provided by Lyft for the demonstration. We combined these two public datasets using the previously proposed knowledge graph-based data integration [9]. After loading the data to the EVOLVE, we have run different object detection tasks using YOLOv4 and CenterNet HourGlass104, EfficientDet D0, and Faster R-CNN ResNet50 V1 models provided from TensorFlow Hub over three different hardware setups. We also conducted a small-scale experiment with the EfficientDet D0 object detection model using transfer learning for detecting traffic lights.

According to the results, EVOLVE allowed the exploitation of much larger knowledge graphs leading to a more efficient application of the object detection models with the help of the knowledge graph-based data integration framework. It significantly improved the execution times compared to running them on a local laptop and a virtual machine.

Eventually, EVOLVE provided the easy-to-use and ready-to-use means to store large datasets and applications of different models with its hardware and software stack.

ACKNOWLEDGMENT

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 825061 (EVOLVE project, www.evolve-h2020.eu). The publication was written at Virtual Vehicle Research GmbH in Graz and partially funded within the COMET K2 Competence Centers for Excellent Technologies from the Austrian Federal Ministry for Climate Action (BMK), the Austrian Federal Ministry for Digital and Economic Affairs (BMDW), the Province of Styria (Dept. 12) and the Styrian Business Promotion Agency (SFG). The Austrian Research Promotion Agency (FFG) has been authorised for the programme management.

REFERENCES

[1] X. Li and B. Zou. An automated data engineering pipeline for anomaly detection of iot sensor data, 2021. [arXiv:2109.13828]
[2] P. Barbiero, G. Squillero, and A. P. Tonda. Modeling generalization in machine learning: A methodological and computational study, 2020. [arXiv:2006.15680]
[3] M. Lenzerini. Data integration: A theoretical perspective. In Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS’02), Madison, Wisconsin, USA, 3-5 June 2002,
[4] T. E. Kalayci, E. Güzel Kalayci, G. Lechner, N. Neuhuber, M. Spitzer, E. Wuestemeier, and A. Stocker. Triangulated investigation of trust in automated driving: Challenges and solution approaches for data integration. Journal of Industrial Information Integration, 21:100186, 2021. doi:10.1007/j.jii.2020.100186

[10] J. Heer and B. Shneiderman. Interactive dynamics for visual analysis. Commun. ACM, 55(4):45–54, April 2012. doi:10.1145/2133806.2137521

[13] Y. Chen, D. Zhao, L. Lv, and Q. Zhang. Multi-task learning for dangerous object detection in autonomous driving. Information Sciences, 432:559–571, 2018. doi:10.1016/j.ins.2017.08.015

[22] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications, 2017. arXiv:1704.04861

[28] M. Tan, R. Pang, and Q. V. Le. Efficiendnet: Scalable and efficient object detection, 2020. arXiv:1911.09070

[29] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection, 2018. arXiv:1708.02947

[34] A. Chazapis, J.-T. Acquaviva, A. Scharl, and D. Soudris. EVOLVE: HPC and cloud enhanced testbed for extracting value from large-scale diverse data. In Proc. of the 18th ACM International Conference on Computing Frontiers, CF’21, page 200–205, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3457388.3458621

[35] C. Tang, N. Srishankar, S. Martin, and M. Tomizuka. Explainable autonomous driving with grounded relational inference. In NeurIPS Machine Learning for Autonomous Driving, 2020. arXiv:2102.11905

[44] A. Tzenetopoulos, D. Masouros, K. Koliogeorg, S. Xydis, D. Soudris, A. Chazapis, C. Kozanitis, A. Bilas, C. Pinto, H. N. Nguyen, S. Louloudakis, G. Gardikis, G. Vamvakas, M. Aabrun, C. Symeonidou, V. Spitidakis, K. Xyliogonapoulos, B. Peischl, T. E. Kalayci, A. Stocker, and J. T. Acquaviva. Evolve: Towards converging big-data, high-performance and cloud computing worlds. In IEEE Design, Automation and Test in Europe Conference, 2022. Accepted.

[45] Computer Architecture and VLSI Systems (CARV) Laboratory. Karvdash — kubernetes carv dashboard. https://github.com/CARV-ICS-FORTH/karvdash, 2021. (accessed on 7 March 2022).

[46] Y. Sfakianakis, C. Kozanitis, C. Kozyrakis, and A. Bilas. QuMan — a kubernetes native object detection framework. In Proc. of the Workshop on Human-In-the-Loop Data Analytics (HILDA’16), San Francisco, California, 26 June 2016, New York, NY, USA, 2016. ACM. doi:10.1145/2939502.2939513

[47] OpenSFS and EOSFS. Lustre® filesystem. https://www.lustre.org/, (accessed on 7 March 2022).

[48] Apache Software Foundation. Apache kafka. https://kafka.apache.org/, 2021. (accessed on 7 March 2022).

[49] Apache Software Foundation. Apache spark. https://spark.apache.org/, 2021. (accessed on 7 March 2022).

[50] A. Tzenetopoulos, D. Masouros, S. Xydis, and D. Soudris. Interference-aware orchestration in kubernetes. In High Performance Computing, 2018. HPCC 2018. The 16th International Conference, pages 667–676, Cham, 2018. Springer International Publishing. doi:10.1007/978-3-030-01521-3_53

[51] D. Elharwi and S. Sakr. Automated machine learning: Techniques and frameworks. In Big Data Management and Analytics, pages 40–69. Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-48277-8_3

[52] M.-A. Zöller and M. F. Huber. Benchmark and survey of automated machine learning frameworks, 2021. arXiv:2101.03970

[53] S. Louloudakis, G. Gardikis, G. Vamvakas, M. Aabrun, C. Symeonidou, V. Spitidakis, K. Xyliogonapoulos, B. Peischl, T. E. Kalayci, A. Stocker, and J. T. Acquaviva. Evolve: Towards converging big-data, high-performance and cloud computing worlds. In IEEE Design, Automation and Test in Europe Conference, 2022. Accepted.
Kalaycı et al.: Object detection in drivingdatasets using a high-performancecomputing platform: A benchmark study
T. E. KALAYCI has been working in Virtual Vehicle Research GmbH, Europe’s largest R&D center for virtual vehicle technology, since 2018, currently as a lead researcher. He received his B.Sc., M.Sc., and Ph.D. from the Computer Engineering Department of Ege University, Turkey. Previously he was an assistant professor at Manisa Celal Bayar University, Turkey, and a postdoctoral researcher at Trento University and Free University of Bozen-Bolzano in Italy. His research interests are data integration, graph databases, intelligent information systems, and machine learning.

G. OZEGOVIĆ has been employed at Virtual Vehicle Research GmbH, Europe’s largest R&D center for virtual vehicle technology, since 2020. She started as a student employee, and is currently working as a junior researcher. She received her bachelor’s degree in Computer Science from Faculty of Engineering Rijeka in Croatia, and is currently finishing her Master’s studies at University of Technology Graz. Her work and studies are focused on data science related topics.

B. BRICELJ holds a master’s degree in economics and finance from University of Maribor’s Faculty of Economics and Business, and a Certificate in Quantitative Risk Management from the International Institute of Professional Education and Research. He worked in the fields of financial services and higher education before transitioning to data science. He has more than five years of experience as a data scientist, implementing statistical, machine learning and deep learning models to analyse and solve various industry specific problems in different industry branches, ranging from automotive industry, heavy industry, to chemical industry. At Virtual Vehicle Research GmbH, he is employed as a Senior Researcher/Data Scientist, working with the “Information Network Extraction Systems” group. His work and research are focused on domains of computer vision and data enrichment.

M. LAH holds an engineering degree in the field of computer science form the University Of Maribor’s Faculty of Electrical Engineering and Computer Science. He started his career in 2009 as a software engineer at Microsoft in the area of Business Intelligence, Data Integration and Analytics. In 2013 he moved from software development to consulting, where he was supporting customers in designing, planning and developing enterprise data-warehouse and analytic solutions. In 2017 he joined Virtual Vehicle Research GmbH, Europe’s largest R&D center for virtual vehicle technology as Lead Researcher and Solutions Architect where he was leading different teams in applied research projects as well as developing production software in the field of modern data integration, knowledge-graphs, machine learning.

A. STOCKER has been a key researcher for contextual information systems and management at Virtual Vehicle Research GmbH, Europe’s largest R&D center for virtual vehicle technology since 2013. Previously, he was key researcher at Joanneum Research, senior researcher at Know-Center, and consultant for information management at Datev. Alexander’s research interests are data-driven information systems and services. He holds a doctorate in business administration from the University of Graz.

***