CCF Transactions on High Performance Computing
https://doi.org/10.1007/s42514-022-00100-4

REGULAR PAPER

EdgeWare: toward extensible and flexible middleware for connected vehicle services

Sidi Lu[1] · Yongtao Yao[1] · Bing Luo[1] · Zhifeng Yu[2] · Dalong Li[2] · Weisong Shi[1]

Received: 24 October 2021 / Accepted: 8 April 2022

Abstract

The dramatic development of Edge Computing technologies is strongly stimulating the adoption of machine learning models on connected and autonomous vehicles (CAVs) so that they can provide a variety of intelligent onboard services. When multiple services running on the resource-constrained CAVs, how limited resources can dynamically support the desired services is of the utmost importance for both automakers and domain researchers. In this context, *efficiently* and *dynamically* managing vehicle services becomes critical for autonomous driving. While previous research focused on service scheduling, computation offloading, and virtual machine migration, we propose EdgeWare, an extensible and flexible middleware to manage the execution of vehicle services, which is open-source to the community with four key features: *i*) on-demand model switch, *i.e.,* easily switch and upgrade machine learning models, *ii*) function consolidation and deduplication to eliminate duplicate copies of repeating functions and maximize the reusability of vehicle services, *iii*) build event-driven applications to reduce workload, and *iv*) dynamic workflow customization which enables customizing workflow to extend the functionality. Our experiment results show that EdgeWare accelerates the execution of services about 2.6 × faster compared to the silo approach and save CPU and memory utilization up to around 50% and 17% respectively, and it allows domain researchers to dynamically add new services on CAVs or easily switch to the upgraded applications for the life cycle management of vehicle services.

Keywords Middleware · Model upgrade · Edge computing

1 Introduction

Connected and autonomous vehicle: The proliferation of communication, robotics, and Edge Computing (Shi et al. 2016) has pushed the horizon of autonomous driving. There has been an acceleration in the research and development (R &D) efforts to bring the idea of connected and autonomous vehicles (CAVs) to fruition. For instance, the advent of Tesla's Autopilot (Gillmore and Tenhundfeld 2020), Google's Waymo (Gibbs 2017), and Baidu's Apollo (Xu et al. 2020) brought CAVs to the spotlight. In the meanwhile, as the perfect Edge Computing platform (Lu et al. 2019), a plethora of intelligent applications are enabled in CAVs such as remote real-time diagnostics (Orf et al. 2020) and advanced driver assistance (Kukkala et al. 2018), which is driven by the enormous vehicle data generated by the equipped multiple sensors such as camera, radar, and LiDAR. These essential components of CAVs are expected to generate around 40 terabytes of data every eight hours of driving, which is the amount of data generated by almost

✉ Sidi Lu
lu.sidi@wayne.edu

Yongtao Yao
yongtaoyao@wayne.edu

Bing Luo
bing@wayne.edu

Zhifeng Yu
zhifeng.f.yu@gmail.com

Dalong Li
dalongli@gmail.com

Weisong Shi
weisong@wayne.edu

[1] Department of Computer Science, Wayne State University, Detroit, MI 48202, USA

[2] Edgemind Solutions LLC, Detroit, MI 48201, USA

Published online: 09 May 2022

3 thousand people (Sidi and Weisong 2021). Moreover, driven by the exponential growth in the usage of CAVs, it is estimated that by 2025, there will be 470 million CAVs on highways worldwide, generating 280 petabytes of data Liu et al. (2020).

Vehicle computing: In the future, more and more data will be generated on CAVs, and it in turn calls for a new computing paradigm called Vehicle Computing (Sidi and Weisong 2021), which refers to the enabling technologies allowing computation to be performed on CAVs. However, different from the data center in the cloud, CAVs nowadays usually have limited computing power, which is mainly hindered by the highly priced hardware system — a level 4 CAV (fully autonomous driving vehicle) can cost up to three hundred thousand dollars, in which sensors and the computing platform cost almost two-thirds of the total price (Sidi and Weisong 2021). Therefore, the computation capability of most CAVs are still limited, and it is necessary to explore how to support multiple computation intensive services for CAVs.

Model upgrading and dependency: The research of Deep Neural Networks (DNNs) has been gaining ever-increasing impetus for autonomous driving due to their state-of-the-art performance. Each year, a multitude of new DNN architectures are proposed Wu et al. (2019) for the emerging intelligent in-vehicle services with more stringent requirements on accuracy improvement, latency reduction, privacy-preserving, and energy efficiency, *etc*. For example, as to camera video processing, R-CNN series (Girshick 2015; Ren et al. 2015; Girshick et al. 2014; He et al. 2017), SSD series (Liu et al. 2016; Fu et al. 2017), and YOLO series (Redmon et al. 2016; Redmon and Farhadi 2017, 2018) are proposed for object detection within two to three years and have been proved to surpass the performance of previous models. This indicates the necessity of dynamic model upgrading for CAV services.

Besides, considering different in-vehicle services, they usually requires the execution of codependent models on the same vehicle computation platform. For instance, in the case of vehicle type identification and vehicle tracking, they both call for the inference results of vehicle detection model. It is an inevitable trend that routine in-vehicle services call for multiple codependent/collaborative deep neural network (DNN) models to finish complicated tasks with remarkable performance (Vanini et al. 2014; Zhou et al. 2015, 2014).

Computation intensive services: However, most of the DNNs focus on boosting accuracy at the expense of substantially increased model complexity — the depth of the current state-of-the-art networks, such as Inceptionv4 (Szegedy et al. 2017) and ResNet-50 (Akiba et al. 2017), can reach dozens or even hundreds of layers to outperformed previous networks for related tasks with accuracy. A single layer may require millions of matrix multiplications. Such heavy calculation brings challenges to deploy these DNN models on a CAV with limited computation resources. Even though CAV computation platforms are becoming increasingly powerful (Dong et al. 2016, 2017), it still falls short when faced with users' growing desire for running more resource-hungry applications, such as intersection analysis Lee et al. (2017) and driver behavior detection Liu et al. (2018). These aforementioned facts make it appealing and crucial to ensure that a range of intelligent services (DNN models) can be deployed and executed *effectively* and *dynamically* a single resource-constrained CAV computation platform and CAV fleets.

Real-world challenges: In the real-world applications, it is common to analyze data streams from different aspects, so researchers usually deploy multiple DNNs on a single CAV working on the same data stream. Take the video analytic as an example, researchers may run the people recognition model in a company surveillance system to identify if the unauthorized people entered a specific area, and at the same time, a behavior detection may also need to detect if there are suspicious actions happened in the surveillance area. Besides, as to autonomous vehicles, they may require accurate lane detection, pedestrian recognition, and so on. As more and more services deployed on CAVs, how shall we dynamically manage and support multiple services becomes a critical problem, which brings more challenges as well as opportunities:

First, how limited resources in CAVs can support the desired services is an open problem. Previous works mainly focus on compressing algorithms by leveraging pruning methods (Cheng et al. 2017; Han et al. 2015), or employ lightweight machine learning packages such as Caffe2, MXNet, and PyTorch. However, no matter how lightweight an algorithm could be, the duplicated calculation (function) among various algorithms still be the bottleneck of the computation resource utilization. Besides, a recent study from Facebook shows that highly heterogeneous devices and low-level software frameworks make it difficult for an application developer to optimize service performance (Wu et al. 2019). We believe that a middleware should be present to deduplicate redundant functions and effectively manage the services so that further optimization is possible just like OpenGL organizes the GPU usage.

Second, how to dynamically support dynamic routine vehicle services is another challenge. In real-world applications, the types and numbers of vehicle services will be changed over time or different application scenarios. For example, surveillance systems could have very different required services from the home to the public areas.

Furthermore, when new on-demand requirements are coming in, extra computation resources are often required and usually affect the routine services on the resource-constrained CAVs. For example, the police may track people's

behavior in real-time as a routine service but may seek a particular person when an emergency happens.

Research gaps: Different from existing approaches, we are not trying to optimize service algorithms or design the service scheduling approaches and offload intensive computations to the remote powerful site since these could be done by application developers. We are also not trying to optimize camera configures, which should be done from the operating system aspect.

Instead, we made the first step towards an extensible and flexible middleware to *dynamically manage the execution, switch, upgrade, and customization of connected vehicle (CV) services*—we propose EdgeWare that encapsulates analysis services in the plug and play function modules and dynamically load or execute them on-demand, and users can choose to distribute and enable necessary modules. We investigate the behavior of multiple modules and observe a new relationship between data and function. We propose function consolidation and deduplication mechanism in EdgeWare that reduces the redundancy in function modules and accelerate the overall data analysis for both real-time edge services. This further enables cooperative analysis among different CAVs. Our case study shows that EdgeWare accelerates execution of services about 2.6× faster compared to silo approach and save CPU and memory utilization up to 50% and 17%, respectively.

Summary of our contributions:

- The extensible and flexible middleware, EdgeWare, is proposed to dynamically manage the execution, switch, upgrade, and customization of vehicle services. It brings a new view of how we deal with vehicle data and vehicle services/functions. Our EdgeWare code used in this study is hosted at https://github.com/GbllYao/EdgeWare.git.
- A simple yet powerful mechanism, function consolidation and deduplication, is designed to effectively integrate different services on CAVs with filtering out redundant functions so that it can save half of the CPU utilization and 17% of memory utilization, and it can speed up the service execution by 2.6× of silo approach.
- The service with event-driven characteristic of EdgeWare involves a trigger followed by a series of DNNs. Only when the trigger was invoked, the later DNNs will be executed. Therefore, building event-driven applications can be used to avoid unnecessary execution of DNNs so that the workload could be further reduced.
- The experiment results of our case study prove that EdgeWare could easily support the on-demand model switch and upgrade, and it could enable customizing workflow to extend the functionality of vehicle services.

The remainder of the paper is as follows. Section 2 elaborates on the motivation of this work. In Sect. 3, we describe the architecture of EdgeWare and discuss how to develop the main techniques that help EdgeWare achieve its goal. Then, we present EdgeWare's case study and its implementation in Sect. 4. Evaluation and discussion are presented in Sect. 5, followed by a discussion of related work in Sect. 6, and finally we conclude the paper in Sect. 7.

2 Motivation

In this section, we seek to answer the following questions: Why intelligent services call for machine learning (ML) model switch and upgrade? Why function consolidation and deduplication is needed for real-world applications? Why event-driven characteristic should be the key feature of vehicle services? Finally, why do we seek to enable dynamic workflow customization?

2.1 Life-cycle management calls for continuous model upgrade

ML Models Degrade with Time: Keeping ML models updated is a necessary step mainly due to the problem of *Concept drift* Tsymbal (2004), which refers to a non-stationary learning problem change over time in unforeseen ways Žliobaitė (2010). The training and the service data often mismatch in real-world problems, and this causes problems because the output of ML models become less accurate as time passes Hand (2006).

New ML models with better performance: Besides, ML models focus on continually boosting accuracy and decreasing computational intensity, *etc.* One such example is the You Only Look Once (YOLO) series (Redmon et al. 2016; Redmon and Farhadi 2017, 2018). In 2015, YOLO was firstly proposed Redmon et al. (2016). Since then, the YOLO series algorithms have been continuously proposed and improved from YOLOv1 Redmon et al. (2016) to the latest version YOLOv4 Bochkovskiy et al. (2020), including the popular light-weight version such as YOLOv3-Tiny Huang et al. (2018).

Diverse data analytic aspects: In addition, it is common to analyze data streams from different aspects with diverse goals. Take autonomous vehicles (AVs) as an example, as the switching between autonomous and manual operation modes, the involved primary computation models are also changed Norris et al. (2011), *e.g.,* from the main computation of lane departure warning and trajectory planning decisions to the traditional stability and reliability-related computation such as tire pressure monitoring and battery failure detection.

Towards this end, some previous work has proposed the concept of continuous learning (Liu 2017; Beneventi et al. 2017) to retrain models and therefore prevent model

degradation without supporting model switching. The most common and direct way is to upgrade ML models. Therefore, it is necessary to build an automated middleware that can support model switching and upgrading for the life-cycle management.

2.2 Maximizing services' reusability needs function deduplication

Redundant function among diverse services: As has been mentioned, multiple services running on the same CAV might have redundant functions (calculations). For example, suppose there are four video-based services including person detection, person tracking, gender classification, and age estimation. Person detection independently detects people for each frame, then the other three services both depend on person detection results to do further analysis. In this context, we can encapsulate each service into a function module and save the person detection module results to the memory so that the other three modules can call the detection results directly and conduct calculations quickly. Therefore, we introduce the idea of *function consolidation and deduplication*, which makes it easy to ship the output of a service for after-the-fact analysis and further computation to reduce the redundant work.

2.3 Workload reduction calls for event-driven applications

Heavy calculations of DNNs: As a subset of ML models, DNNs are currently gaining high momentum in industry and academia due to its state-of-the-art performance on previously-thought hard problems, but most of the DNNs focus on boosting accuracy at the expense of substantially increased model complexity, and such heavy calculation brings challenges to deploy even a single DNN model on a resource-constrained vehicle device. Besides, running multiple DNNs at the same time usually leads to high memory and CPU Utilization. Application developers are struggling with the limited memory bandwidth of CAVs that have to be used to store the huge amounts of weights and activations in DNNs, and high CPU usage might damage the processor or other components of CAV computation unit.

Avoid unnecessary execution of DNNs: On the other hand, it is not necessary to always execute DNNs for a specific application since traditional ML models can have equal performance or beat DNNs under a specific application scenario (Lu et al. 2020). Following these insights, EdgeWare focuses on building the event-driven service, which involves a trigger followed by a series of DNNs. Only when the trigger was invoked, the later DNNs will be executed. Therefore, developing event-driven services can be used to avoid unnecessary execution of DNNs so that the workload could be reduced.

2.4 Task change requires dynamic workflow customization

The need to extend workflow functionality: In the real-world applications, the service scenarios are often changed and therefore the categories of the corresponding tasks are usually changed or accumulated over time. One such example is the drone. The detection tasks of a drone might change from face detection only to both the vehicle and face detection when it flys from the high foot traffic regions to a highway area. Besides, since these service requests are sometimes unpredictable, it is hard for developers to design a fixed workflow for the first time. Hence, it is necessary to enable customizing workflow to extend functionality for vehicle services.

3 EdgeWare architecture

In this section, we introduce EdgeWare's architecture and describe how to develop the main techniques that help EdgeWare achieve its goals. Specifically, EdgeWare has been specifically conceived with the following goals in mind:

(i) *On-demand model switch and upgrade*, to deal with the life-cycle management of intelligent vehicle services;
(ii) *Function consolidation and deduplicaton*, to encapsulate each service into a function module so that it is easy to ship the output of a single service for further analysis and reduce the redundant calculations;
(iii) *Event-driven applications*, to invoke DNNs in response to external events and reduce unnecessary computation.
(iv) *Dynamic workflow customization*, to allow customizing workflow and extend service functionality.

3.1 Function consolidation and deduplication

Observations and basic idea: One of the key advantages of Edge Computing is less data movement over the network, which can also be regarded as moving functions (calculations) to the proximity of the data source. In this paper, we push this idea further by introducing "function consolidation and deduplication" for the vehicle service management, which is motivated by the fact that when a variety of analysis services are running on CAVs, many of them share the same sub-tasks that are independent to other parts of services. Take the video analysis as an example, both face recognition and people behavior detection require person

detection. In this case, we can conduct person detection only once, then save and feed the result to the other two services. Therefore, to maximize the reusability of vehicle services and use the computing resources of the CAVs more effectively, we encapsulate each service into a function module that can plug and play on CAVs. In this way, we can save and directly ship the redundant output of a function for the fast after-the-fact analysis and further computation without redundant calculations. The idea of "function consolidation and deduplication" originated from the "data cache" (Nesbit and Smith 2004; Jiang et al. 2019) that has been widely used in different levels of computer design. By putting frequently used data in a higher speed storage medium for faster reuse, data cache speeds up the overall computing performance. With the same principle, we consolidate function to achieve function deduplication and faster reuse, so as to accelerate the whole process. We call this process as *function consolidation and deduplication*, and we list three advantages as follows.

- **Maximize the reusability of vehicle services:** When multiple services are running on CAVs, many of them share the same sub-tasks that are independent of other parts of services, *e.g.,* both face recognition and people behavior detection require person detection. In this case, we can conduct person detection only once and feed the detection results to these two services. Figure 1 presents an example of function consolidation and deduplication and illustrates the basic idea. In the silo approach, suppose module X and Y contains two functions respectively. Since module X and module Y share function b, we can implement function module a, b, and c so that module b only need to execute once and other modules can also leverage the results of module b as well.

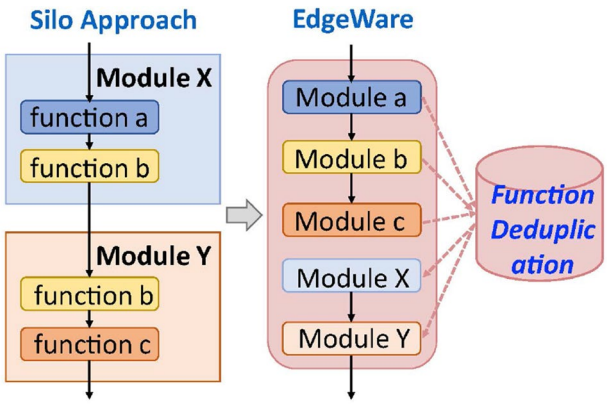

Fig. 1 An example of function consolidation and deduplication. We encapsulate each service into a function module and consolidate functions for faster reuse

- **Enable faster vehicle services:** Second, when a service is triggered by a user, *function consolidation* will search for results in the cache and directly return available results to the proper module for the fast analytics, without waiting for the results of redundant services. For example, the police officer may do people detection as a routing service and cache the detection results accordingly since many services are based on it, such as person recognition and person tracking. It is not necessary to conduct face recognition all the time, but if the defined events happen, they can easily search in the cache to find out all people appeared quickly and only do face matching for these people. Therefore, by eliminating the redundancy of in-vehicle models, the overall performance of the CAV computation unit will definitely be improved if such overlap exists, and more vehicle services running on the same platform creates a larger possibility to deduplicate function executions.
- **Facilitate collaborative analysis:** Besides, the function consolidation and deduplication makes it easy to do collaborative analysis. On the same device, any module can conduct analysis base on previous results from either the same module or other modules. When multiple and even heterogeneous CAVs are involved, any module on CAVs can query information from other devices. In this case, the advantages of function consolidation and deduplication becomes more obvious.

3.2 EdgeWare design

Flogo Framework: EdgeWare is built on top of the open-source project—Flow-based process engine written in Go (Flogo) Project Flogo (2016) from TIBCO Software Inc. Flogo is a Go-powered and lightweight Edge Computing ecosystem for building event-driven applications, *i.e.,* leveraging triggers and actions to process incoming events. It is implemented in Go programming language and therefore it is 20~50× more lightweight than similar Java or JavaScript frameworks.

Definition of flow: The EdgeWare application is composed of one or more Flows, and each Flow is composed of a trigger and multiple activities. The activities can be connected in series or combined into multiple branches, as shown in Fig. 2. The concept and pattern of Flow in Flogo perfectly fit our original intention of designing EdgeWare, so we chose to develop EdgeWare on the basis of Flogo.

Application architecture: The application architecture of EdgeWare is shown in Fig. 3. Here, we describe Fig. 3 from bottom to top.

– **Flogo core** is the application kernel, which provides developers the necessary components to build event-driven services.

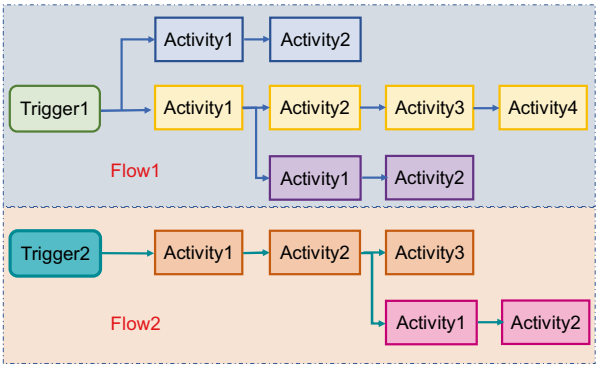

Fig. 2 The examples of flows defined in this work

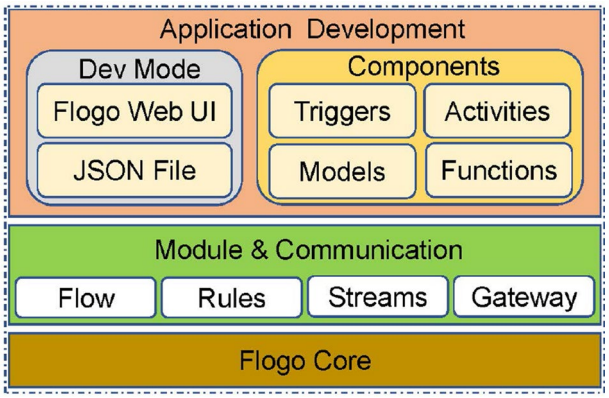

Fig. 3 Application architecture of EdgeWare

- **Flow** provides application integration capabilities, allowing developers to connect incoming events with applications, databases, and APIs.
- **Rules** simplify the complexity of real-time contextual decision making. Due to the large number of events generated from various sources, it can be very valuable to understand these events in a given context, so that developers could use a series of rules to detect patterns and trends in order to take appropriate actions.
- **Streams**: developers could adopt, aggregate, and preprocess the event stream to generate a single derived event, in which measures can be taken to provide a clean and simple way to process the data stream.
- **Gateway**: gateway patterns could be implemented flexibly, and developers are able to enforce policies, restrict or route traffic conditionally, and use data from another endpoint to enrich requests.
- **Flogo web UI**: to develop a EdgeWare application, the Flogo web user interface (Web UI) is a good choice. Developers can also directly define the components of a program and the corresponding connection and communication through a JSON file.

Limitations of flogo: As an open-source ecosystem, Flogo allows developing ultra-lightweight vehicle applications and also provides many basic activities and triggers, such as MQTT, WebSockets, CoaP, and REST Showcase of awesome activities (2016). Flogo platform claimed that it can provide zero-coding web user interface since the application development methods are like building blocks where activities are triggers are like blocks; however, such application development has two obvious limitations as follows.

- In order to develop applications suitable for specific scenarios and specific needs, developers are required to use Go programming language to implement triggers and activities to achieve specific goals,
- Flogo supports the TensorFlow model inference but does not support TensorFlow model training.
- Flogo currently only supports the ML framework of TensorFlow 1.X, and it does not contain any TensorFlow-related libraries.

3.3 EdgeWare implementation

Environment configuration of EdgeWare: As a prerequisite for the development of EdgeWare, Flogo currently only supports the ML framework of TensorFlow 1.X. Besides, as a Go-based ultra-lightweight open source ecosystem, Flogo is mainly used to build event-driven applications, and it does not contain any TensorFlow-related libraries. So in EdgeWare, the TensorFlow dynamic library must be installed on both the developer's device and the user's device, and the dynamic library must be built specifically for the platform architecture, such as Linux Arm, x86, x64, Darwin, *etc.* After successfully setting the library path or related environment variables, the EdgeWare development environment is ready. Besides, considering the real-world application, most of EdgeWare's modules should be DNN-based, but Flogo is not very friendly to DNN support. For example, Flogo supports the TensorFlow model inference, but it does not support model training. As to EdgeWare, we should use Python to train the model first, and export the model in the SavedModel format for running and inference in Flogo.

Code 1 Common Interface

```
function INIT( )
    //Create an activity register.
end function

function NEW(ctx activity.InitContext)
    //New optional factory method, should be used if one
    //activity instance per configuration is desired.
end function

function METADATA(*activity.Metadata)
    // Returns the metadata of the activity.
end function

function EVAL(ctx activity.Context)
    //Eval is called when an Activity is being evaluated.
    //Returning true indicates that the task is done.
end function
```

Build EdgeWare services: Flogo provides a common interface enabling developers to build a trigger or activity. More specifically, taking the implementation of an activity as an example, developers need to implement these functions as shown in Code 1 when building an activity. In addition to implementing the functions defined by the common interface, the received and passed data of the activity should also be specified by defining variable's type and name. In Code 1, the Eval function is the main body of activity, and the inference of DNN is done by calling this function. To conduct DNN's inference, in addition to converting the DNN model to the SavedModel format, it is also necessary to obtain the input tensor name and output tensor name of the DNN model. In this way, the EdgeWare application can input data into TensorFlow Graph and obtain the result after the inference is completed. In addition, the image data must be resized to the input size of the model and converted to the tensor format.

Goals achieved: Here, we describe our solutions to the four goals presented at the beginning of Sect. 3.

- **Solution to on-demand model switch and upgrade**: EdgeWare can achieve easily model switch and upgrade without compiling. Specifically, it uses the REST (REpresentational State Transfer) Battle and Benson (2008) trigger allowing users to pass parameters to the EdgeWare runtime through HTTP (HyperText Transfer Protocol) requests. The parameters will be used to update the variable values during EdgeWare runtime, and the variables will be passed to every related module through Flows. The update of the variable value will cause the DNN model to be reloaded. If a highly accurate model is available, it can be loaded into the EdgeWare during runtime, *i.e.,* users can switch and upgrade models without stopping the current EdgeWare service.
- **Solution to function consolidation and deduplication:** As has been mentioned, a Flow is composed of multiple activities, which can be connected in series or combined into branches. An activity will transmit the same data to all activities connected behind it. In EdgeWare, when a piece of data needs to be used in multiple activities, the data is only cached once and shared by other associated activities. To save the storage space, there is no backup in the cache, and data will not be frequently transmitted multiple times, thus saving bandwidth. In addition, take the Flow1 in Fig. 2 as an example, the yellow Activity1 is connected to the yellow Activity2 and the purple Activity1, which means that a function in the yellow Activity1 is executed only once and will be passed to these two branches behind it. On the contrary, if the yellow and purple branches are split into two Flows, all functions in Activity1 are duplicated. Therefore, the structure of Flows in Fig. 2 realizes function consolidation and deduplication.
- **Solution to build event-driven services**: Event-driven refers to a strategy for making decisions in the continuous activity management, *i.e.,* based on the events that occur at the certain time point, to mobilize available resources and perform related tasks so that emerging problems can be resolved. The event-driven architecture usually uses loose coupling, because the event initiator does not know which user is listening to the current event, and it does not know the subsequent results. Since event-driven architecture can minimize the degree of coupling, it is an ideal choice for modern distributed application architecture. EdgeWare is developed based on Flogo, and event-driven is one of the core concepts of Flogo. Take the Flow as an example. It consists of a trigger and multiple activities. Once the trigger is invoked, all activities in the entire Flow will execute their Eval function (shown in Code 1).
- **Solution to dynamic flow customization**: The coupling between different Flows of EdgeWare is very low or even without coupling, which makes dynamic Flow customization easy to achieve. We also use the HTTP requests to pass a value of a parameter to the EdgeWare runtime. When the transmitted value is equal to the trigger condition of a certain flow, the later activities will be invoked. Assuming that EdgeWare contains n Flows, and they are divided into two groups (Group A Flows and Group B Flows). All triggers of Group A Flows are controlled via the same parameter value, so does Group B Flows but its parameter value is different. As such, we can achieve dynamic Flow customization by controlling the execution of different groups' Flows for diverse application scenarios.

4 Raspberry Pi based video analysis: a case study

In this section, we will show how EdgeWare could be used in vehicle video analysis. We trained and deployed five ML models including motion detection, face detection, gender classification, age estimation, and emotion detection on the Raspberry Pi, and treat it as the case study to evaluate the performance of EdgeWare compared with the silo approach. Note that although both our design and implementation are exclusively for video analysis, our idea can be easily adapt to any stream data analysis like audio or any digital sensor data.

4.1 Five involved ML models

Go-based motion detection: Motion detection is a very essential task for many computer vision tasks especially in video surveillance systems analysis. Motion detection refers to detecting changes in the position of an object relative to its surrounding environment or changes in the surrounding environment relative to the object Elharrouss et al. (2019). In EdgeWare, the motion detection method is the traditional machine learning methods Motion Detection (2017), *i.e.,* threshold-based motion detection algorithms, which computes the pixel-wise difference between two continuous video frames and output if a kind of motion is detected. Since traditional ML models could achieve comparable accuracy and dominates the motion detection field (Parks and Fels 2008; Kumar and Sureshkumar 2013), we do not leverage any computation intensive DNNs for motion detection.

Go-based face detection: Face detection is a popular topic in the computer vision area and has received significant research progress in terms of detection accuracy (Viola and Jones 2004; Jiang and Learned-Miller 2017). However, most of the face detection approaches are powered by the developer-friendly languages like Python (Zhang et al. 2016) considering the language community, the number of available libraries, and the simplicity of the language, *etc.*, and few of them are written in the Golang programming language which is usually faster than Python and consumes less memory. Since the core of EdgeWare is based on the Go-powered Flogo framework, we implement a Go-based face detection model called Go-face by using the face recognition libraries for Golang Face Detection (2018).

Go-based gender classification: Recognizing human gender has become very popular recently because intelligent services with this feature can provide a more user-friendly environment for specific genders and social interactions (Gupta 2015; Kim et al. 2006; Ahmed and Kabir 2012). We implement a Go-powered gender classification model based on the TensorFlow version model Gender Classification (2017) which first calls dlib library to detect and align faces in images and then uses a convolutional neural network (CNN) to classify gender based on the detected human faces.

Go-based age estimation: Automatic estimation of human age through facial image analysis has many potential practical applications, such as human-computer interaction and multimedia communication (Rothe et al. 2015, 2018). Inspired by the work of Gender Classification (2017), which uses the FaceNet architecture Schroff et al. (2015) and is pre-trained on ImageNet for image classification and feature extraction. Fine-tuning the CNN on training images with apparent age annotations is a necessary step, so we trained the age estimation model based on the IMDB-WIKI dataset which consists of 524,230 face images crawled from IMDB and Wikipedia websites, and we implemented a Go-based age estimation model.

Go-based emotion detection: The face is a characteristic feature of human since it contains identity and emotion information. It is possible to identify a person and her/his characteristics such as emotion (or expression) according to her/his face images. Based on Facial detection, recognition and emotion detection (2019), which are implemented by TensorFlow and working for facial detection, recognition, and emotion detection, we implemented a Go version and use it for the emotion detection which categorizes seven types of emotion, including angry, sad, neutral, disgust, surprise, fear, and happy.

4.2 Four public datasets

In order to identify the effectiveness of EdgeWare in terms of function duplication on the face related applications, we collect and analysis four public datasets that provides annotations for gender and age classifications.

Adience benchmark of unfiltered faces for gender and age classification dataset: In order to facilitate the study of age and gender recognition, the Open University of Israel provides a dataset and benchmark of face photos (Levi and Hassner 2015; Eidinger et al. 2014). The images contained in this dataset is designed to be as realistic as possible to the real-world imaging challenges. In particular, it tries to capture all changes in appearance, pose, lighting, *etc.* that can be expected of images taken without careful preparation or posing. The image source of this dataset is assembled by automatic upload from iPhone5 (or later) smartphones, and released to the public by their authors under the Creative Commons (CC) license Hassner et al. (2015).

IMDB-WIKI dataset: The IMDB-WIKI dataset is the largest publicly available training dataset of face images

with gender and age annotations, which is published by ETH Zurich. Based on the list of the most popular 100,000 celebrities shown on the IMDb website, ETH Zurich crawled their public information related to the date of birth, name, gender, and all images (Rothe et al. 2015, 2018). In addition, ETH Zurich used the same meta-information to retrieve all profile images from Wikipedia's people page, and deleted images without a timestamp (the date the photo was taken). Assuming that images of a single face are both likely to show an actor, and the time stamp and the birth date of that actor are both correct, then they can assign an actual age to each image. Besides the incorrect timestamps, many images are still images from movies. In total, they obtained 460,723 facial images from 20,284 celebrities from IMDb and 62,328 celebrities from Wikipedia, resulting in a total of 523,051 images.

UTKFace dataset: The UTKFace dataset Zhang et al. (2017) is a large-scale face dataset with a large age range (spanning from 0 to 116 years old). The dataset contains more than 20,000 facial images with age, gender, and ethnicity annotations. The images cover big changes in posture, facial expressions, lighting, occlusion, resolution, *etc.* This dataset can be used for various tasks, such as face detection, age estimation, age classification, landmark positioning, *etc.*

AppaReal dataset: The AppaReal dataset Clapés et al. (2018) released by the University of Barcelona, which contains 7591 images and related real age annotations. The total number of apparent votes was approximately 250,000. On average, there are about 38 votes per image, which makes the average apparent age very stable (0.3 standard errors of the average). The images are divided into 4113 training images, 1500 validation images and 1978 testing images.

4.3 Model training platform

Based on the above four public datasets, we trained ML models on the NVIDIA GPU Workstation, which is capable to deliver the cluster-level performance for even the demanding applications (Spiga and Girotto 2012; Morozov et al. 2011). NVIDIA GPU Workstationis equipped with a powerful Intel Xeon E5-2690 v4 CPU and four GeForce RTX 2080 Ti GPUs. The memory of it is 64 GB and the operating system installed is Windows 10 containing 14 cores. After training, we deploy these ML models on the Raspberry Pi to test the performance of EdgeWare.

4.4 Implementation of the case study

Description of the case study diagramming: Figure 4 shows the diagramming of our case study application, including six Flows that consists of diverse triggers and activities. Each Flow is responsible for a specific task with the corresponding trigger. We mark these Flows as

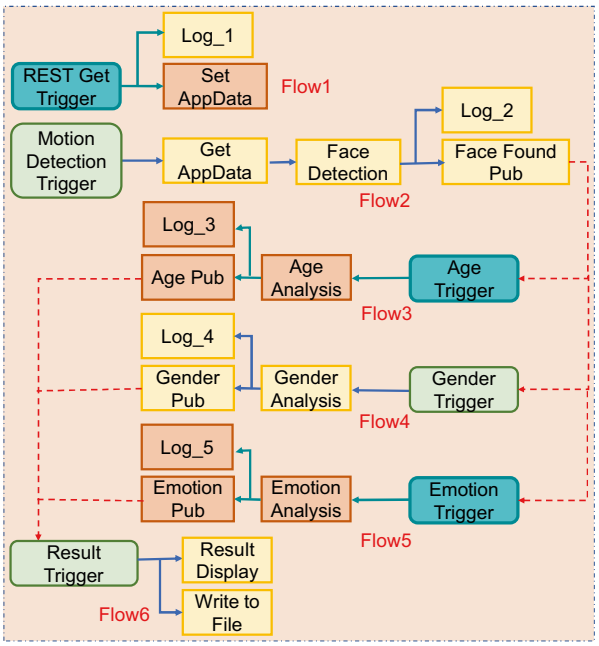

Fig. 4 Implementation of the case study

Flow1~Flow6 from top to bottom of Fig. 4. The Log_Number (*e.g.,* Log_1) in Fig. 4 represents the corresponding data or the message in a specific Flow. The usage of Log_Numbers does not affect the function of the application but helps us develop an application.

In Fig. 4, the REST Trigger of Flow1 defines two methods — Get and Post, which provides the EdgeWare application with the ability to start an action via REST over HTTP. REST refers to a set of constraints that ensure a fault-tolerant, scalable, and extendible system. To be concrete, we can use the following command in the terminal to pass parameters to the EdgeWare runtime: curl http://localhost:8020/demoID/3. Here, both the port and name are configurable.

Flow1 (Pass Data to All Flows): In order to pass the data obtained by EdgeWare in the REST trigger to all Flows, we applied the Set method of AppData activity in Flow1 (shown in Fig. 4). Note that this activity defines the Set and Get methods that allow users to set and get global App attributes, which means that Other Flows can obtain the demoID data originally from the curl command through the Get method of AppData activity. The Get method of AppData activity is not called repeatedly in multiple Flows, and We only call this method in Flow2 (*i.e.,* Get AppData). Note that the output of Face Found Pub activity in Flow2 (face detection) will be used as the trigger of Flow3 (age estimation), Flow4 (gender classification), and Flow5 (emotion detection). We wrap the demoID data obtained by the Get method of AppData activity in an event message and pass it to the next three Flows.

Flow2 (Face Detection): In Flow2, Motion Detection Trigger is used to read the video stream of the USB camera

and conduct a motion detection algorithm to detect the difference of every two continuous frames. If the difference is exceeding the threshold, it will trigger the subsequent activities. On the other hand, Face Detection calls the dlib library and DNN model to detect if the incoming frame contains a face or not, if it does, the face area is cropped and saved into the cache. At the same time, Face Found Pub will post a message telling that it has detected the face and the cached location of the face image. The Face Found Pub is a Kafka publisher. Here, **Kafka** is a distributed messaging system for collecting and delivering high volumes of log data with low latency (Kreps et al. 2011). Specifically, Kafka is in general publish-subscribe based messaging system, which consists of the publisher and subscriber. Publisher publishes messages and subscriber consume or pull that data. In this work, Kafka publisher and subscriber are the core components in the case study. They are the communication hub of Flow2~Flow6. We use Kafka publisher and subscriber to solve the communication problem of Flow2~Flow6. Thanks to the distributed division of Kafka, EdgeWare applications can be easily deployed in a distributed environment.

Code 2 Invoke DNN Model

```
1: function EVAL(ctx activity.Context)
2:     //Image preprocessing to obtain input tensor.
3:     result, err := model.Session.Run(
4:     map[tf.Output]*tf.Tensor{
5:     model.Graph.Operation("input_1").Output(0): imgtf,},
6:     []tf.Output{
7:     model.Graph.Operation("dense/Softmax").Output(0),
8:     }, nil,
9:     )
10: end function
```

Flow3 (Age Estimation), Flow4 (Gender Classification), and Flow5 (Emotion Detection): Flow3, Flow4, and Flow5 are very similar. The difference is that different DNN models are invoked. These DNN models are used to predict age, gender, and emotions. As has been mentioned in Sect. 3, in order to invoke the DNN model, the input and output tensor names of the model need to be explicitly specified, as shown in Code 2, "input_1" and "dense/Softmax" are the input and output tensor names respectively. All of the three Flows here invoke DNN models in this way, which is the only TensorFlow 1.x calling method currently supported by Flogo. The triggers of these three Flows are all Kafka subscribers, they listen to the same event from Flow2. The last activity of the three Flows is a Kafka Publisher, they publish event messages to the same Kafka Topic, which are monitored by the trigger (a Kafka subscriber) of Flow6.

Flow6 (Receive, Integrate, and Display Results): Flow6 has nothing to do with DNN. It just receives results from the previous Flows, *i.e.,* gender classification, age estimation, and emotion detection results, then integrate, displays, and writes them to the file.

Figure 5 presents an output result of our case study. The left side is the original video frame, and the right side is the result of the cropping face area and results of gender classification, age estimation, and emotion detection, *i.e.,* the result of a frame of picture input from Flow2 and finally output from Flow6.

5 Evaluation and discussion

In this section, we evaluate the performance of EdgeWare in the case study of video analysis with diverse ML models, including face detection, gender classification, age estimation, and emotion detection. Specifically, our evaluation answers the following questions:

a) Can EdgeWare switch and upgrade machine learning models? (answered in Sect. 5.3)
b) How well does EdgeWare eliminate duplicate functions and maximize the reusability of vehicle services? (answered in Sects. 5.2.2 and 5.2.3)

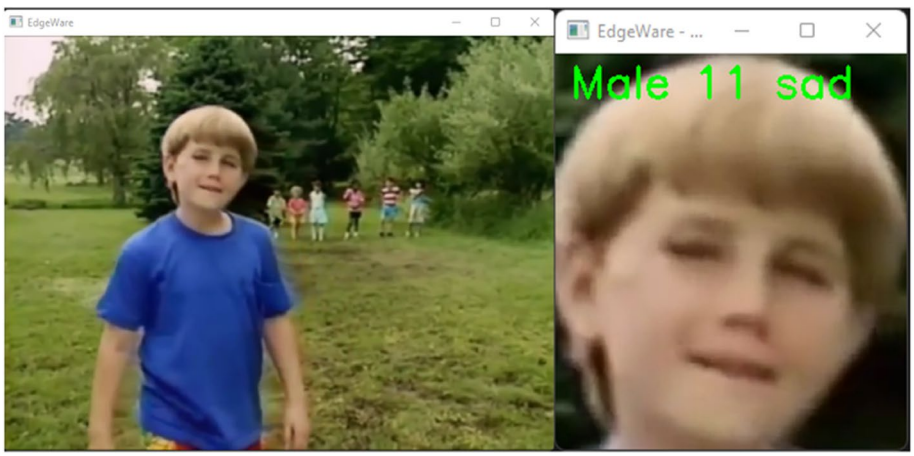

Fig. 5 Case study demo

Fig. 6 A raspberry Pi 4B and a USB camera

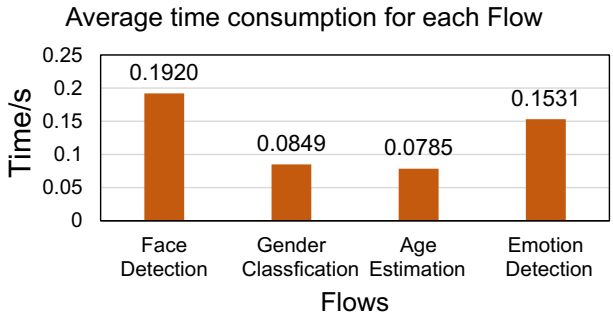

Fig. 7 Average time consumption of processing an image for each Flow in the silo approach

c) How long inference time does EdgeWare can reduce? (answered in Sect. 5.2.1)
d) How well does EdgeWare's dynamic Flow customization can enable the customizing of Flow to extend the functionality? (answered in Sect. 5.3)

5.1 Hardware setup

In this case study, we adopt two types of hardware in total to test the performance of EdgeWare, *i.e.,* a Raspberry Pi 4B and a USB camera (shown in Fig. 6). The edge node (CAV) is Raspberry Pi 4B, which is a micro-computer running on a Linux system. Raspberry Pi 4B is the latest version released in June 2019 BinMasoud and Cheng (2019), and it can communicate with the Internet via an on-board Wi-Fi module. Besides, it is a Broadcom BCM2711 system-on-chip with a quad-core Cortex-A72 (ARM v8) 64-bit processor running at 1.5 GHz, with 4 GB of RAM Danish et al. (2020).

In addition, we use the ELP mini USB Camera to capture real-world video for the evaluation purpose. The USB camera can be directly used to connect with any personal computer and all kinds of OS system machine by USB 2.0's Standard-A port for security and computer vision analysis, with HD resolution up to 1080×720P.

5.2 Experimental observation

Next, we test the performance of real-time service using EdgeWare compared with a silo approach, in terms of inference time, CPU and memory consumption, and answer the proposed questions in Sect. 5. In a silo approach, four flows *i.e.,* face detection, gender classification, age estimation, and emotion detection are executing independently. Whereas using EdgeWare, face detection is the first module needed to be executed, then the other three modules receive the face detection results to conduct corresponding calculations concurrently.

5.2.1 Execution time comparison

Figure 7 presents the average execution time of each module for a single frame in the silo approach. It can be seen that each frame of face detection consumes the longest time on average, which is reasonable since it needs to conduct intensive pixel-wise calculation, while the other three services, *i.e.,* gender classification, age estimation, and emotion detection can be regarded as the classification problems. Suppose we use T_{silo} and $T_{EdgeWare}$ to represent the overall execution time of silo approach and EdgeWare services, and use T_{face}, T_{gen}, T_{age}, and T_{emo} to denote the execution time of each module respectively. Therefore, the formulas to calculate T_{silo} and $T_{EdgeWare}$ can be defined as follows.

$$T_{silo} = \left(T_{face} + T_{gen}\right) + \left(T_{face} + T_{age}\right) + \left(T_{face} + T_{emo}\right) \quad (1)$$

$$T_{EdgeWare} = T_{face} + max\left\{T_{gen} + T_{age} + T_{emo}\right\} \quad (2)$$

Based on Eqs. 1 and 2, we have $T_{silo} = 0.8925(s)$ and $T_{EdgeWare} = 0.3457(s)$, *i.e.,* the overall execution time of silo approach is around 0.89 s and the inference time of EdgeWare service is about 0.35 s, which answers the question of "how long inference time does EdgeWare can reduce" at the beginning of Sect. 5.

Observation 1 EdgeWare approach is about 2.6× faster than the silo approach in this particular scenario.

5.2.2 CPU utilization comparison

Figure 8 presents the comparison results of the EdgeWare approach and the silo approach in terms of CPU utilization. We use blue color and orange color to denote the sum of work handled by a CPU of EdgeWare and silo approach, respectively. It can be seen that the CPU utilization of EdgeWare is normally half of the silo approach. More specifically, on average, EdgeWare services account for around 95.96% CPU, while the CPU utilization of the silo method is 193.04%. It proves that using EdgeWare could reduce the CPU usage for our case study. Since the CPU utilization can reflect the workload status, we can infer that EdgeWare could reduce the computation

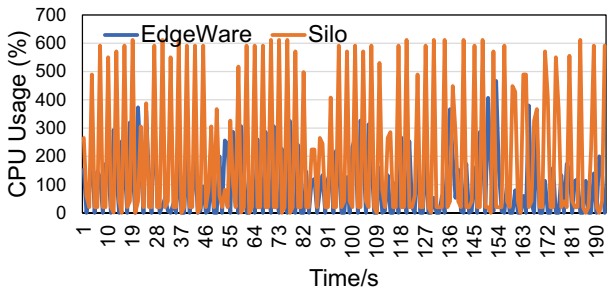

Fig. 8 CPU usage comparison between EdgeWare and silo approach

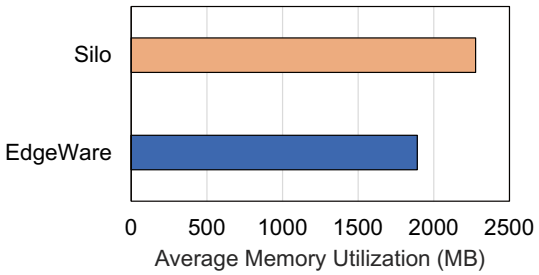

Fig. 9 Average memory utilization comparison between EdgeWare and silo approach

workload by eliminating the redundant calculations, which answers the question of "how well does EdgeWare eliminate duplicate functions and maximize the reusability of vehicle services" at the beginning of Sect. 5 from one aspect.

Observation 2 EdgeWare services can save half of the CPU utilization compared with silo methods in our case study.

5.2.3 Memory utilization comparison

With the same principle, we also measure the memory utilization of the EdgeWare application and silo approach, which is shown in Fig. 9. The average memory utilization of EdgeWare services is about 1890.35 MB, and the value of the silo approach is around 2276.22 MB. Similarly, it also answers the question of "how well does EdgeWare eliminate duplicate functions and maximize the reusability of vehicle services" at the beginning of Sect. 5 from the memory usage aspect. The comparison with silo approach is able to reflect the performance of EdgeWare. We conducted experiments in Raspberry Pi, and in addition to the execution of these flows, there is also competition for computing resources from the associated environment. When flows are distributed and deployed in different nodes, EdgeWare can support distributed deployment of multiple models and enable cooperation among multiple edge devices.

Observation 3 On average, the memory utilization of EdgeWare services is lower than the silo approach by 17% in our case study.

5.3 Evaluation on dynamic flow customization and model upgrade

Three flows with different accuracy-level models: In order to answers the two questions of "*a*) Can EdgeWare switch and upgrade machine learning models?" and "*d*) How well does EdgeWare's dynamic flow customization can enable the customizing of flow to extend functionality?", we evaluate three Flows, *i.e.,* gender classification, age estimation, and emotion detection. Besides, as to gender classification, we have low-accuracy and high-accuracy DNN model. Similarly, we also consider two different accuracy-level DNN models for age estimation. Figure 10 present an example

Fig. 10 An example of Flow customization and model upgrading

of Flow customization (*i.e.,* the capability of adding new Flows for other intelligent services) and model upgrade (*i.e.,* switching from the low-accuracy model to the high-accuracy model).

To be concrete, Fig. 10a represents the gender classification Flow with low-accuracy model; Fig. 10b denotes the gender classification Flow and the age estimation Flow, and these two Flows both include low-accuracy models; Fig. 10c shows the results of three Flows, *i.e.,* gender classification, age estimation, and emotion detection, with high-accuracy models. Comparing these subfigures, we can see that using EdgeWare, we can improve the accuracy of gender classification by ugrade the low-accurate model to the high-accurate model, *i.e.,* the gender classification result of Fig. 10a and b are both wrong while we can eliminate this false classification by using high-accurate model (Fig. 10c) Besides, we can also add new services in the EdgeWare approach accordingly.

To better demonstrate the uniqueness and innovation of EdgeWare, we list below the main differences between EdgeWare and Flogo.

- To apply a neural network model to Flogo and deploy it on a Raspberry Pi-like device, we needs to convert the model format and get the neural network node names, which EdgeWare provides the tools for.
- Function consolidation and deduplication is the core idea of EdgeWare, based on the communication mechanism of ZooKeeper and Kafka, which makes cross-platform function consolidation and deduplication easy.
- The model update mechanism is designed by ourselves, combined with Pub/Sub communication mode, passing configuration information through JSON, synchronizing the model update status notification to all flows, and realizing the adjustment of related activities under the new model at the same time.

6 Related work

6.1 Standalone service in single-platform

Previous studies on Edge Computing have created a variety of video analysis services for applications in the fields of public safety, autonomous vehicles (AVs), emergency medical service, *etc.* Shi et al. (2016). However, running such services on the resource-constrained edge is very challenging since video and image processing are both bandwidth-hungry and computationally intensive. Towards this end, in vehicular computing, Lu *et al.* proposed an edge-based object detection services for autonomous vehicles on the optical-domain compressed video Lu et al. (2020), with the primary goal of accelerating accurate video analysis and decreasing energy consumption. Hossein Badri *et al.* proposed a service placement method optimized for edge services that take edge environment into consideration Badri et al. (2017). Peng Liu *et al.* designed a high-level, task-specific API for the user to apply Deep Neural Networks by transforming models trained with popular deep learning frameworks Liu et al. (2018). Arun Ravindran and Anjus George designed a key-value edge data store for video analysis, which monitors run-time conditions to feed data to latency-critical tasks to improve the overall quality Ravindran and George (2018).

6.2 Cloud/edge service management

Service scheduling. Zhang *et al.* analyze the bandwidth, energy, and computing power of using cloud/edge computing and proposed EVAPS (Edge Video Analysis for Public Safety) to leverage various computing platforms to achieve higher efficiency for public safety Zhang et al. (2016). Chien-Chun Hung *et al.* presented a system that tries to figure out the best Video analytics query plan among cameras, private clusters, and public clouds to achieve a good trade-off between computation resources and algorithm accuracy Hung et al. (2018). Similarly, Lee *et al.* described a methodology for scheduling interactions initiated by vehicular applications and measured both the computational and network demands of offloading analysis to edge infrastructure Lee et al. (2017).

Computation offloading. One of the most popular service management studies in Edge Computing is computation offloading. The fundamental idea is to leverage powerful remote resources by offloading complex computations to the remote site. Dong *et al.* developed two offloading algorithms as a resource management framework to determine which task components should be offloaded to optimize the response time while minimizing energy consumption Dong et al. (2017). Yi *et al.* proposed LAVEA that provides various task placement schemes and formulates an optimization problem for offloading task selection to provide low-latency video analytics at places closer to the users Yi et al. (2017). Drolia *et al.* introduced a prefetching and caching technique to selectively run part of computation on edge devices so that it can reduce the need to offload images to the cloud Drolia et al. (2017). Wang *et al.* described four real-time video analytics strategies for drones to reduce total transmission and save bandwidth while minimally impact result accuracy and latency. Wang et al. (2018). Luo et al. (2021) conducted a comprehensive review of resource scheduling in edge computing. Arthurs et al. (2021) surveyed the literature for connected vehicles and provided taxonomies for their use cases. Luo et al. (2020) proposed reinforcement learning based vehicular edge computing.

Vehicle services management. Wang et al. (2019) proposed a prototype system for distributed task scheduling on mobile edge devices, and they aimed to enhance on-board Vehicle Computing Units for CAVs. Besides, they also proposed three scheduling strategies. Their work focused on the dynamic management, resource monitoring, and task offloading of the edge devices for the scheduling platform. However, this scheduling prototype is based on socket communication and has poor scalability. Zhao and Kim (2020) argued that in general scenarios where there are multiple cloud/edge devices and competing resources, it is important to consider vehicle trajectories, workloads, and requests simultaneously to jointly optimize allocations. They investigated the cost minimization problem in the allocation and scheduling of interconnected vehicle service requests on heterogeneous cloud/edge services. However, whether there is redundancy among multiple tasks and whether these requests can be optimized are not considered.

Virtual machine migration. In the meanwhile, techniques based on virtual machine migration have been proposed in Kiryong et al. (2015), Satyanarayanan et al. (2009) to accelerate the service handoff across edge servers. Ma *et al.* designed a service handoff system that migrates services to the nearest edge server by Docker layered storage system migration to reduce file system synchronization overhead Ma et al. (2017). Jang *et al.* proposed a task offloading mechanism for efficient video analytics processing and created virtualized Docker containers Merkel (2014) with a dynamic reconfiguration scheme that allows IoT cameras to dynamically adjust their configuration to environmental context changes without degrading application QoS (Jang et al. 2018).

6.3 Comparison with rocket

Rocket video analytics system: In the end of 2019, Microsoft has proposed Rocket Microsoft Rocket for Live Video Analytics (2019), which is a powerful configurable platform for live video analytics. The platform works across geographically distributed CAVs (*e.g.,* Azure Stack Edge) and large clouds (*e.g.,* Azure Machine Learning and Cognitive Services), and its ultimate goal is to make it easy and affordable for real-time, low-cost and accurate live video analysis.

The need of cloud-agnostic CAV computing platforms: The services of Microsoft Rocket are tightly coupled with their respective cloud platform services. There is a need for cloud-agnostic, platform-agnostic Edge Computing platforms that customers can deploy on-prem. While the project Flogo from TIBCO is a lightweight vehicle computing platform that is not tied to any specific public cloud platforms. Inspired by Flogo, we build EdgeWare toward extensible and flexible middelware, specifically for the vehicle service management.

Similarity between rocket and EdgeWare: Rocket and EdgeWare introduce the concept of "pipeline" and "Flows" respectively, which both includes a cascade of ML models for a specific service, and the pipline or Flow can both be customized. Developers can also augment the above pipelines or Flows with simpler motion detection filters based on OpenCV background subtraction. Besides, both Rocket and EdgeWare are able to be configured to execute over a distributed infrastructure.

Difference between EdgeWare and rocket: In general, EdgeWare is different from Rocket in terms of six aspects, which is shown in Table 1. Rocket is a platform for live video analytics. With perfect integration with Azure cloud services as its feature, and the powerful computing and storage capabilities in the cloud make Rocket much more powerful. Rocket is developed for Windows and is hard to be deployed on Raspberry Pi, which is not very compatible with edge devices.

- EdgeWare is an Go-powered and ultra-light platform while Rocket is built on *C#* which has robust base class libraries.
- The Flows of EdgeWare can be easily customized since the coupling between different Flows is very low or even without coupling. As to Rocket, although developers can also customize the pipeline architecture of Rocket, the

Table 1 Comparison between EdgeWare and rocket

EdgeWare	Rocket
Go	C#
Flows (Easy to customize)	Pipelines (Can be customized, but more difficult)
Flexible model switching	Recompile
Support Raspberry Pi	Visual Studio Project on Windows Linux: docker + NVIDIA driver
Based on Edge platform only	Edge + Cloud platform
Provide models and model switch/update services	No model switch/update services
Only TensorFlow 1.x is supported	Support for different DNN frameworks
Not support query results	Support query results

customizing process is more difficult due to the high coupling of diverse pipelines.
- EdgeWare can achieve flexible model switching without recompiling, while Rocked requires recompiling to realize model switching.
- EdgeWare focuses on low-cost edge devices with machine learning capabilities, such as Raspberry Pi. Rocket supports Visual Studio projects on the Windows operating system, while in the Linux operating system, Rocket requires docker and NVIDIA drivers to run, which makes Rocket not applicable to many edge devices.
- EdgeWare is a lightweight edge computing platform that is not tied to any specific public cloud platform, instead, it only focuses on edge platform so that developers can deploy on-prem. However, Rocket is tightly coupled with its cloud platform services *e.g.,* Azure Machine Learning and Cognitive Services.
- EdgeWare provides ML models and model switch/update services, while Rocket does not provide this kind of services.
- As to EdgeWare, only TensorFlow 1.x is supported, while Rocket supports different ML model framework, such as TensorFlow, Darknet or Caffe.
- EdgeWare does not support query results, while Rocket does.

6.4 Distinctive requirements to design vehicle service middleware

Although CAVs are the perfect and typical Edge Computing platforms in the Edge Computing era Lu et al. (2019), the design of vehicle service middleware should take into account the distinctive characteristics of vehicle services, especially considering the communication and collaboration of vehicle fleets in the real-world application scenario. In this subsection, we list two unique characteristics of EdgeWare for connected vehicle services.

First, EdgeWare is able to kill in-vehicle services with a time or an event trigger, and this function is necessary since complete in-vehicle services are not always needed in the real-world application scenario. Take the intersection management in a CAV environment as an example, at each intersection, CAV fleets and several roadside units (RSUs) are working together to coordinate the movement of all vehicles. Intelligent RSUs can observe and predict the trajectories of all moving vehicles based on traffic detection devices (such as radar or LiDAR-based sensors) and equipped motion detection algorithms Lin et al. (2017). In this case, if it is predicted that an approaching vehicle is going to leave the specific intersection area, EdgeWare can receive the signal (trigger) and kill in-vehicle services (such as traffic light detection and front vehicle tracking) even if these services are not finished. Second, EdgeWare is able to share/send the intermediate service results (such as pedestrian detection and trajectory prediction results) to the surrounding vehicles, and surrounding vehicles could continue to conduct data analysis based on the received intermediate results. This ability could save the computation resources and reduce inference time especially when multiple vehicles are working together.

7 Conclusion remarks

In this work, we propose a framework, EdgeWare, towards extensible and flexible middleware for vehicle services, which has four key features: *i*) on-demand model switch, *i.e.,* easily switch and upgrade machine learning models, *ii*) function consolidation and deduplication to eliminate duplicate copies of repeating functions and maximize the reusability of vehicle services, *iii*) build event-driven applications to reduce workload, and *iv*) dynamic workflow customization which enables customizing workflow to extend the functionality. Our experiment results show that EdgeWare has the capability to accelerate the overall data analysis performance for real-time services, and it allows researchers and application developers to dynamically add new services on CAVs or easily switch to the upgraded applications for the life cycle management of vehicle services. This work is supported in part by the National Science Foundation (NSF) grants CNS-2140346.

References

Ahmed, F., Kabir, M.H.: Facial feature representation with directional ternary pattern (dtp): Application to gender classification. In: 2012 IEEE 13th International conference on information reuse & integration (IRI), pp. 159–164 (2012). IEEE

Akiba, T., Suzuki, S., Fukuda, K.: Extremely large minibatch sgd: training resnet-50 on imagenet in 15 minutes. arXiv:1711.04325 (2017)

Arthurs, P., Gillam, L., Krause, P., Wang, N., Halder, K., Mouzakitis, A.: A taxonomy and survey of edge cloud computing for intelligent transportation systems and connected vehicles. IEEE Trans. Intell. Transp. Syst. (2021)

Badri, H., Bahreini, T., Grosu, D., Yang, K.: Multi-stage stochastic programming for service placement in edge computing systems: Poster. In: Proceedings of the Second ACM/IEEE Symposium on Edge Computing. SEC '17, pp. 28–1282. ACM, New York, NY, USA (2017). https://doi.org/10.1145/3132211.3132461

Battle, R., Benson, E.: Bridging the semantic web and web 2.0 with representational state transfer (rest). J. Web Semant. **6**(1), 61–69 (2008)

Beneventi, F., Bartolini, A., Cavazzoni, C., Benini, L.: Continuous learning of hpc infrastructure models using big data analytics and in-memory processing tools. In: Design, automation & test in

Europe Conference & Exhibition (DATE), 2017, pp. 1038–1043 (2017). IEEE
BinMasoud, A., Cheng, Q.: Design of an iot-based vehicle state monitoring system using raspberry pi. In: 2019 International Conference on Electrical Engineering Research & Practice (ICEERP), pp. 1–6 (2019). IEEE
Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M.: YOLOv4: optimal speed and accuracy of object detection. arXiv:2004.10934 (2020)
Cheng, Y., Wang, D., Zhou, P., Zhang, T.: A survey of model compression and acceleration for deep neural networks. arXiv:1710.09282 (2017)
Clapés, A., Bilici, O., Temirova, D., Avots, E., Anbarjafari, G., Escalera, S.: From apparent to real age: gender, age, ethnic, makeup, and expression bias analysis in real age estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 2373–2382 (2018)
Danish, M., Brazauskas, J., Bricheno, R., Lewis, I., Mortier, R.: Deepdish: multi-object tracking with an off-the-shelf raspberry pi. In: Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking, pp. 37–42 (2020)
Dong, Z., Gu, Y., Chen, J., Tang, S., He, T., Liu, C.: Enabling predictable wireless data collection in severe energy harvesting environments. In: 2016 IEEE Real-Time Systems Symposium (RTSS), pp. 157–166 (2016). IEEE
Dong, Z., Gu, Y., Fu, L., Chen, J., He, T., Liu, C.: Athome: Automatic tunable wireless charging for smart home. In: Proceedings of the Second International Conference on Internet-of-Things Design and Implementation, pp. 133–143 (2017)
Dong, Z., Liu, Y., Zhou, H., Xiao, X., Gu, Y., Zhang, L., Liu, C.: An energy-efficient offloading framework with predictable temporal correctness. In: Proceedings of the Second ACM/IEEE Symposium on Edge Computing. SEC '17, pp. 19–11912. ACM, New York, NY, USA (2017). https://doi.org/10.1145/3132211.3134448
Drolia, U., Guo, K., Narasimhan, P.: Precog: Prefetching for image recognition applications at the edge. In: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, pp. 1–13 (2017)
Eidinger, E., Enbar, R., Hassner, T.: Age and gender estimation of unfiltered faces. IEEE Trans. Inf. Forensic. Secur. **9**(12), 2170–2179 (2014)
Elharrouss, O., Al-Maadeed, N., Al-Maadeed, S.: Video summarization based on motion detection for surveillance systems. In: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), pp. 366–371 (2019). IEEE
Face Detection. https://github.com/Kagami/go-face (2018)
Facial detection, recognition and emotion detection. https://github.com/priya-dwivedi/face_and_emotion_detection/blob/master/Facial%20Detection%2C%20Recognition%20and%20Emotion%20Detection.md (2019)
Fu, C.-Y., Liu, W., Ranga, A., Tyagi, A., Berg, A.C.: Dssd: Deconvolutional single shot detector. arXiv:1701.06659 (2017)
Gender Classification. https://github.com/BoyuanJiang/Age-Gender-Estimate-TF (2017)
Gibbs, S.: Google sibling waymo launches fully autonomous ride-hailing service. The Guardian **7**, (2017)
Gillmore, S., Tenhundfeld, N.L.: The good, the bad, and the ugly: Evaluating tesla's human factors in the wild west of self-driving cars. In: Human Factors and Ergonomics Society Annual Meeting (2020)
Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014)
Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp. 1440–1448 (2015)
Gupta, S.: Gender detection using machine learning techniques and delaunay triangulation. Int. J. Comput. Appl. **124**(6) (2015)
Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv:1510.00149 (2015)
Hand, D.J.: Classifier technology and the illusion of progress. Stat. Sci. 1–14 (2006)
Hassner, T., Harel, S., Paz, E., Enbar, R.: Effective face frontalization in unconstrained images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4295–4304 (2015)
He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2961–2969 (2017)
Huang, R., Pedoeem, J., Chen, C.: YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers. In: 2018 IEEE International Conference on Big Data (Big Data), pp. 2503–2510 (2018). IEEE
Hung, C., Ananthanarayanan, G., Bodik, P., Golubchik, L., Yu, M., Bahl, P., Philipose, M.: VideoEdge: Processing camera streams using hierarchical clusters. In: 2018 IEEE/ACM Symposium on Edge Computing (SEC), pp. 115–131 (2018). https://doi.org/10.1109/SEC.2018.00016
Jang, S.Y., Lee, Y., Shin, B., Lee, D.: Application-aware iot camera virtualization for video analytics edge computing. In: 2018 IEEE/ACM Symposium on Edge Computing (SEC), pp. 132–144 (2018). https://doi.org/10.1109/SEC.2018.00017
Jiang, H., Learned-Miller, E.: Face detection with the faster r-cnn. In: 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), pp. 650–657 (2017). IEEE
Jiang, B., Yang, J., Ding, G., Wang, H.: Cyber-physical security design in multimedia data cache resource allocation for industrial networks. IEEE Trans. Ind. Inf. **15**(12), 6472–6480 (2019)
Kim, H.-C., Kim, D., Ghahramani, Z., Bang, S.Y.: Appearance-based gender classification with gaussian processes. Pattern Recognit. Lett. **27**(6), 618–626 (2006)
Kiryong, H., Yoshihisa, A., Zhuo, C., Wenlu, H., Brandon, A.: Adaptive vm handoff across cloudlets. technical report cmu-cs-15–113. Computer Science Department, Carnegie Mellon University (2015)
Kreps, J., Narkhede, N., Rao, J., et al: Kafka: A distributed messaging system for log processing. In: Proceedings of the NetDB, vol. 11, pp. 1–7 (2011)
Kukkala, V.K., Tunnell, J., Pasricha, S., Bradley, T.: Advanced driver-assistance systems: a path toward autonomous vehicles. IEEE Consumer Electron. Magn. **7**(5), 18–25 (2018)
Kumar, A.N., Sureshkumar, C.: Background subtraction based on threshold detection using modified k-means algorithm. In: 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering, pp. 378–382 (2013). IEEE
Lee, K., Flinn, J., Noble, B.D.: Gremlin: Scheduling interactions in vehicular computing. In: Proceedings of the Second ACM/IEEE Symposium on Edge Computing. SEC '17, pp. 4–1413. ACM, New York, NY (2017). https://doi.org/10.1145/3132211.3134450
Levi, G., Hassner, T.: Age and gender classification using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 34–42 (2015)
Lin, P., Liu, J., Jin, P.J., Ran, B.: Autonomous vehicle-intersection coordination method in a connected vehicle environment. IEEE Intell. Transp. Syst. Magz. **9**(4), 37–47 (2017)
Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European Conference on Computer Vision, pp. 21–37 (2016). Springer
Liu, L., Lu, S., Zhong, R., Wu, B., Yao, Y., Zhang, Q., Shi, W.: Computing systems for autonomous driving: state-of-the-art and challenges. IEEE Int. Things J. (2020)

Liu, P., Qi, B., Banerjee, S.: Edgeeye: An edge service framework for real-time intelligent video analytics. In: Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking. EdgeSys'18, pp. 1–6. ACM, New York, NY, USA (2018). https://doi.org/10.1145/3213344.3213345

Liu, L., Qiao, X.Z.M., Shi, W.: Safeshareride: Edge-based attack detection in ridesharing services. In: USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18). USENIX Association, Boston, MA (2018). https://www.usenix.org/conference/hotedge18/presentation/liu

Liu, B.: Lifelong machine learning: a paradigm for continuous learning. Front. Comput. Sci. **11**(3), 359–361 (2017)

Lu, S., Luo, B., Patel, T., Yao, Y., Tiwari, D., Shi, W.: Making disk failure predictions smarter! In: 18th USENIX conference on file and storage technologies (FAST' 20), pp. 151–167 (2020)

Lu, S., Yao, Y., Shi, W.: Collaborative learning on the edges: A case study on connected vehicles. In: 2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19) (2019)

Lu, S., Yuan, X., Shi, W.: An integrated framework for compressive imaging processing on cavs. In: the Fifth ACM/IEEE Symposium on Edge Computing (SEC '20). IEEE, Virtual (2020)

Luo, Q., Hu, S., Li, C., Li, G., Shi, W.: Resource scheduling in edge computing: A survey. IEEE Commun. Surv. Tutor. (2021)

Luo, Q., Li, C., Luan, T.H., Shi, W.: Collaborative data scheduling for vehicular edge computing via deep reinforcement learning. IEEE Int. Things J. **7**(10), 9637–9650 (2020)

Ma, L., Yi, S., Li, Q.: Efficient service handoff across edge servers via docker container migration. In: Proceedings of the Second ACM/IEEE Symposium on Edge Computing. SEC '17, pp. 11–11113. ACM, New York, NY, USA (2017). https://doi.org/10.1145/3132211.3134460

Merkel, D.: Docker: lightweight linux containers for consistent development and deployment. Linux J. **2014**(239), 2 (2014)

Microsoft Rocket for Live Video Analytics. https://www.microsoft.com/en-us/research/project/live-video-analytics/ (2019)

Morozov, I.V., Kazennov, A., Bystryi, R., Norman, G.E., Pisarev, V., Stegailov, V.V.: Molecular dynamics simulations of the relaxation processes in the condensed matter on gpus. Comput. Phys. Commun. **182**(9), 1974–1978 (2011)

Motion Detection. https://github.com/hybridgroup/gocv/blob/release/cmd/motion-detect/main.go (2017)

Nesbit, K.J., Smith, J.E.: Data cache prefetching using a global history buffer. In: 10th International symposium on high performance computer architecture (HPCA'04), pp. 96–96 (2004). IEEE

Norris, W.R., Allard, J., Filippov, M.O., Haun, R.D., Turner, C.D.G., Gilbertson, S., Norby, A.J.: Systems and methods for switching between autonomous and manual operation of a vehicle. Google Patents. US Patent 7,894,951 (2011)

Orf, S., Zofka, M.R., Zöllner, J.M.: From level four to five: Getting rid of the safety driver with diagnostics in autonomous driving. In: 2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), pp. 19–25 (2020). IEEE

Parks, D.H., Fels, S.S.: Evaluation of background subtraction algorithms with post-processing. In: 2008 IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance, pp. 192–199 (2008). IEEE

Project Flogo. https://github.com/TIBCOSoftware/flogo (2016)

Ravindran, A., George, A.: An edge datastore architecture for latency-critical distributed machine vision applications. In: USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18). USENIX Association, Boston, MA (2018). https://www.usenix.org/conference/hotedge18/presentation/ravindran

Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)

Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7263–7271 (2017)

Redmon, J., Farhadi, A.: YOLOv3: An incremental improvement. arXiv:1804.02767 (2018)

Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)

Rothe, R., Timofte, R., Van Gool, L.: DEX: Deep expectation of apparent age from a single image. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 10–15 (2015)

Rothe, R., Timofte, R., Van Gool, L.: Deep expectation of real and apparent age from a single image without facial landmarks. Int. J. Comput. Vis. **126**(2–4), 144–157 (2018)

Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The case for VM-based cloudlets in mobile computing. IEEE Pervas Comput **8**(4), 14–23 (2009)

Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 815–823 (2015)

Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Int. Things J. **3**(5), 637–646 (2016)

Showcase of awesome activities, triggers and apps for Flogo. https://tibcosoftware.github.io/flogo/showcases/ (2016)

Sidi, L., Weisong, S.: The emergence of vehicle computing. IEEE Int. Comput. (2021)

Spiga, F., Girotto, I.: phigemm: a cpu-gpu library for porting quantum espresso on hybrid systems. In: 2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing, pp. 368–375 (2012). IEEE

Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: Thirty-first AAAI Conference on Artificial Intelligence (2017)

Tsymbal, A.: The problem of concept drift: definitions and related work. Comput. Sci. Dept Trinity Coll. Dublin **106**(2), 58 (2004)

Vanini, Z.S., Khorasani, K., Meskin, N.: Fault detection and isolation of a dual spool gas turbine engine using dynamic neural networks and multiple model approach. Inf. Sci. **259**, 234–251 (2014)

Viola, P., Jones, M.J.: Robust real-time face detection. Int. J. Comput. Vis. **57**(2), 137–154 (2004)

Wang, J., Feng, Z., Chen, Z., George, S., Bala, M., Pillai, P., Yang, S.-W., Satyanarayanan, M.: Bandwidth-efficient live video analytics for drones via edge computing. In: 2018 IEEE/ACM Symposium on Edge Computing (SEC), pp. 159–173 (2018). IEEE

Wang, L., Zhang, Q., Li, Y., Zhong, H., Shi, W.: Mobileedge: Enhancing on-board vehicle computing units using mobile edges for cavs. In: 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), pp. 470–479 (2019). IEEE

Wu, C.-J., Brooks, D., Chen, K., Chen, D., Choudhury, S., Dukhan, M., Hazelwood, K., Isaac, E., Jia, Y., Jia, B., et al: Machine learning at Facebook: Understanding inference at the edge. In: 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 331–344 (2019). IEEE

Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 10734–10742 (2019)

Xu, K., Xiao, X., Miao, J., Luo, Q.: Data driven prediction architecture for autonomous driving and its application on apollo platform. In: 2020 IEEE Intelligent Vehicles Symposium (IV), pp. 175–181 (2020). IEEE

Yi, S., Hao, Z., Zhang, Q., Zhang, Q., Shi, W., Li, Q.: Lavea: Latency-aware video analytics on edge computing platform. In: Proceedings of the Second ACM/IEEE Symposium on Edge Computing. SEC '17, pp. 15–11513. ACM, New York, NY, USA (2017). https://doi.org/10.1145/3132211.3134459

Zhang, Z., Song, Y., Qi, H.: Age progression/regression by conditional adversarial autoencoder. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017). IEEE

Zhang, Q., Yu, Z., Shi, W., Zhong, H.: EVAPS: Edge video analysis for public safety. In: 2016 IEEE/ACM Symposium on Edge Computing (SEC), pp. 121–122 (2016). https://doi.org/10.1109/SEC.2016.30

Zhang, K., Zhang, Z., Li, Z., Qiao, Y.: Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Process. Lett. **23**(10), 1499–1503 (2016)

Zhao, Y., Kim, B.: Optimizing allocation and scheduling of connected vehicle service requests in cloud/edge computing. In: 2020 IEEE 13th International Conference on Cloud Computing (CLOUD), pp. 361–369 (2020). https://doi.org/10.1109/CLOUD49709.2020.00057

Zhou, P., Dai, L., Jiang, H.: Sequence training of multiple deep neural networks for better performance and faster training speed. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5627–5631 (2014). IEEE

Zhou, P., Jiang, H., Dai, L.-R., Hu, Y., Liu, Q.-F.: State-clustering based multiple deep neural networks modeling approach for speech recognition. IEEE/ACM Trans. Audio Speech Lang. Process. **23**(4), 631–642 (2015)

Žliobaitė, I.: Learning under concept drift: an overview. arXiv:1010.4784 (2010)