A Cloud-based Continual Learning System for Road Sign Classification in Autonomous Driving

Charalampos Davalas, Dimitrios Michail, Christos Diou, Iraklis Varlamis and Konstantinos Tserpes

Harokopio University of Athens, 17778, Athens

Abstract

Artificial intelligence and deep learning have demonstrated highly promising results in challenging problems, such as autonomous or assisted driving. One challenge in the integration of these solutions in real-life applications, is that they often operate in a resource-constrained edge environment. Another important challenge is the ability of the AI system to adapt and expand its abilities in a constantly changing environment. Constant changes could potentially cause significant deterioration of a model's effectiveness, a phenomenon called *Catastrophic Forgetting*. In this paper, we propose a *Continual Learning framework* for efficient and continuous update of a road sign classification system for assisted or autonomous driving. Our proposition considers the limitations of edge computing and utilizes a cloud infrastructure. Test results show that the our proposition is capable of expanding an edge models knowledge in a stable manner.

Keywords

Machine Learning, Supervised Learning, Continual Learning, Catastrophic Forgetting

1. Introduction

Safe and effective operation of an autonomous vehicle assumes that its driving mechanism perceives the environment including nearby vehicles, pedestrians and other obstacles, traffic and road signs, as well as the driver condition. Using this information, the vehicle can take action automatically, or hand-off control to the human pilot. Pervasive AI can be beneficial in multiple autonomous driving scenarios, can assist human drivers and at the same time can learn from their feedback. In this work we assume the scenario of a traffic sign detection module that continuously provides feedback to the autonomous driving system and the driver about the signs that appear along a route, in various conditions. The heart of this module is an online image classifier, which is trained, using as feedback the reaction of the driver to its classification decisions. For the implementation of the module we assume a cloud-edge setup, in which a model on the edge is used for inference. The model is periodically retrained on the cloud using i) a limited amount of data that is being transferred from the edge to the cloud, and ii) additional data that reside on the cloud that are used for repairing and controlling the model performance in order to avoid catastrophic forgetting [1].

For simplicity, we assume that the various traffic signs correspond to three major groups (which we call "action classes") that ask for a different behavior from the driver (or the auto-pilot): i) to stop the vehicle, ii) to decelerate, iii) to keep a constant speed. Consequently, the classifier output can either be directly transformed to an action for the auto-pilot or become a recommendation for the driver. Following the training approach that we described above it is also possible for a generic traffic sign detection module to adapt to the specific driver's preferences, to keep being trained on new, unknown or noisy signs, and even to learn new tasks and their respective signs.

This simplified example and architecture that is depicted in Figure 1 can be expanded to more complex cases, in which a pre-trained decision module can evaluate a situation (by analyzing sensor data on the edge) and suggest an action to the driver (or take the action directly). At the same time, it can take advantage of the driver's feedback and keep improving the decision module. The architecture balances between the limited memory and computational resources of the edge, which are used for input stream processing and inference, and the unrestricted cloud resources that are used to periodically re-train the model and avoid any drifts or performance decays.

2. Related Work

In our previous work in [2] we combined a rehearsal method for mitigating catastrophic forgetting [1] in a resource-constrained, online setup while reducing training times and minimizing both memory and computational requirements. This work is mostly a proof of con-

Proceedings of the 1st International Workshop on Computational Intelligence for Process Mining (CI4PM) and the 1st International Workshop on Pervasive Artificial Intelligence (PAI), co-located with the IEEE World Congress on Computational Intelligence (WCCI), Padua, Italy, 18–23 July 2022

cdavalas@hua.gr (C. Davalas); michail@hua.gr (D. Michail); cdiou@hua.gr (C. Diou); varlamis@hua.gr (I. Varlamis); tserpes@hua.gr (K. Tserpes)

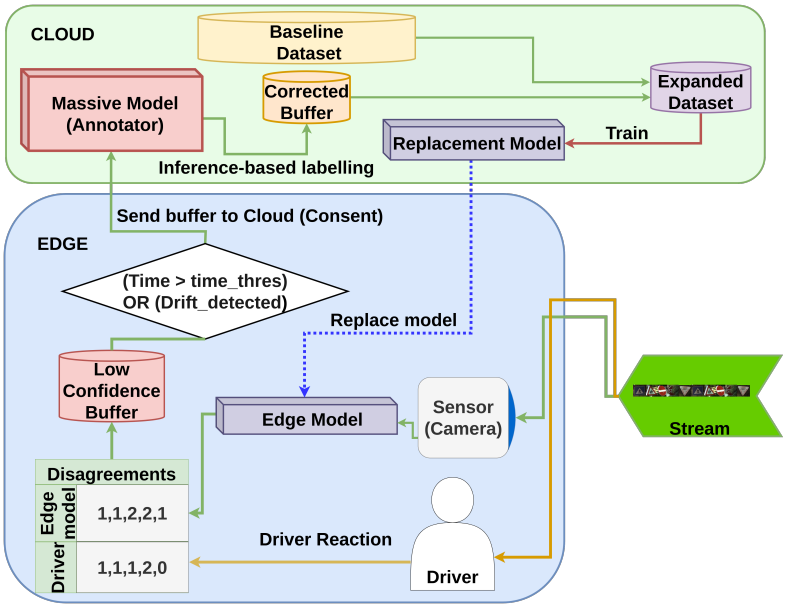


Figure 1: Overview of the proposed cloud-edge architecture for continual learning with traffic signs.

cept and we specifically use an artificial stream of data, albeit without any focus on autonomous driving. Another approach is the use of Echo State Networks (ESN) [3] for the purposes of continual learning in streaming scenarios with Reservoir Computing.

In the bibliography, most approaches that focus on computational efficiency in resource constrained environments do not consider model updates using new data. In [4], a setup of two lightweight networks is used for the purposes of traffic sign classification at the edge with the help of knowledge distillation. The work of [5] focuses on efficient detection of traffic signs in a road setup but does not include a continual learning process. In [6] a complete smart city setup is proposed that uses image detection and classification in IoT devices, however it does not include provisions for frequent or continuous model updates.

3. Traffic sign classification on the edge

3.1. The autonomous vehicle scenario

As demonstrated in Figure 1 the traffic sign classification scenario includes an autonomous vehicle, which is a equipped with a forward-facing camera that sees what the driver sees, an object detection module that is responsible for detecting traffic signs from the camera input in real-time and a pre-trained image classification module that assigns a detected sign to one of the three action classes (i.e. stop, decelerate, keep constant speed). Based on the output of the classification module a different action is recommended to the driver. The vehicle is also equipped with a set of sensors that record the driver's reaction and a second classifier that assigns the driver's reaction into one of the three action classes.

The classification module performance is periodically evaluated using as input the driver's reaction and the classifier output and any disagreements are recorded at the edge.

When significant deviations are detected between the two decisions, the system triggers a cloud-based model update. In this case, the recorded cases of disagreement are transmitted to the cloud, where the original model is retrained. The retraining process employs additional samples (traffic sign images) in order to avoid catastrophic forgetting. The updated model is fed back to the edge and the same loop goes on continuously.

3.2. Edge Setup

As mentioned before, the edge device is used only for processing the camera input and detecting the signs, and for classifying the sign accordingly using the pre-trained model. It also communicates with the cloud for sending the cases of disagreement, i.e the new signs that have been detected and classified but differently than the driver's response.

In the following, we assume excellent performance of

the module that interprets the driver's reaction and the sign detection module and we focus on the performance of the traffic sign classifier. In order to simulate the operation of the classifier in a driving scenario, we create a stream of traffic signs and feed them to the pre-trained model. The stream of signs can take various shapes, from a random selection of images from all "action classes" to the more demanding setup of non-i.i.d images. For the simulation of the driver's experience we assume that the human driver correctly guesses the label with a high probability.

A major threat for online learning systems is the gradual or sudden changes in the distribution of samples, or the appearance of new samples from different classes or tasks. This is called *concept drift* [7] and can be handled by early detection of the change and retraining of the model, or by constantly retraining the model with some offline or past knowledge. A *Drift detector* is a mechanism that can assist in the former case, by providing an alert when two output sources are diverging according to a specific statistical model. These changes can be abrupt or gradual depending on the context and there are different drift detectors depending on the situation [8]. The drift detector can be adjusted to check at periodic time intervals and activate a drift signal only when disagreements occur. In our scenario we use the ECDD drift detector [9] due to its simplicity and its ability to detect changes quickly.

4. Model re-training on the cloud

Complementing the edge setup that is shown in Figure 1 is the cloud infrastructure, where the retraining of the image classification model takes place. Each time the driver disagrees with the output of the classifier on the edge, the respective sign images are stored in a buffer. In this setup the drift detector uses both the driver reactions and the model inference in order to keep statistics of the disagreements between the driver and the model. When the drift occurs, the drift activator sends the alarm signal. When the drift detector raises an alert, the buffer is being transferred to the cloud where a more complicated deep learning model, uses them for further training. Note that we do not use the driver actions as labels due to the aforementioned disagreements.

The large model on the cloud, also referred as the annotator, is used only for inference and plays the role of an oracle for the sign images that arrive from the edge. It has already been trained with a large labeled dataset that comprises an adequate variety of images and has a much higher complexity than the model running on the edge (more layers and nodes per layer). Thus, the large model could not be trained on the edge due to edge nodes' hardware limitations. For this purpose, another simpler model, with the same architecture as the one running on the edge is trained on the cloud, using as input the correctly labeled images transferred from the edge and a dataset which contains an adequate amount of refined samples (baseline). The training samples are shuffled and fed in batches in order to update the small model on the cloud. The updated model is then used to replace the previous edge model.

5. Experimental setup

5.1. Dataset

Our dataset is actually a customization of the Belgium traffic signs [10] dataset. We split a subset of approximately 1900 images from the original dataset in two groups. The first, *Baseline* group *B*, comprises approximately half of the images that are good quality and are used as a basis for pretraining and/or repairing the edge model on the cloud.

The *Stream* group, *S* comprises the remaining images which include lower quality images that may appear with angle distortion, low-light, graffiti or discoloration. These are used to populate the *stream* of images that arrive to the edge model. In both cases the baseline and stream datasets have been selected manually to simulate the difference between high quality images that are used for model training, vs the lower quality images that often appear in practice.

Three labels (0,1 and 2) have been assigned to each image, that correspond to the correct action. Label 0 corresponds to no action, allowing the driver to continue with their current driving status, label 1 is deceleration without stopping, and in label 2 the car must stall as shown in Figure 2.

In order to evaluate we sample a small number of images (about 100) in held out datasets from both the Stream and Baseline group. The evaluation by stream images can indicate the model's ability to learn from the stream data, while the evaluation from the baseline portion can give as a insight of how well a model can maintain its former knowledge. In both cases the test images have been held out, i.e. are not used in any way during training.

When navigating a road network, sign classes are not "shuffled" e.g., one may frequently observe stop signs (images with label 2) in an urban neighborhood however these are rarely observed in high-speed roads. To simulate this we split the stream into "tasks", each containing data from two classes only. The stream therefore consists of consecutive blocks of 30 to 50 batches each, with every batch containing 16 traffic sign images. Each block corresponds to one task only, thus the first block includes images with labels in $\{0, 1\}$. Similarly the second and third blocks include images with labels in $\{1, 2\}$ and $\{2, 0\}$

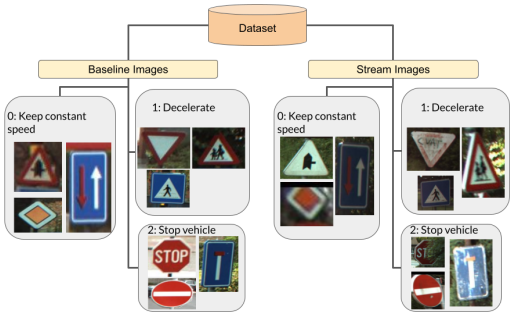


Figure 2: Overview of the dataset.

respectively. This sequence of tasks repeats in a cyclic manner.

5.2. Models

Training a model inside a vehicle is a challenging issue due to the limitations that might apply in terms of computational power and memory capacity.

Due to these conditions we chose to balance between edge and cloud by using a very light network comprised of one VGG block [11] at the edge for inference and small scale training (if any). For the annotator, we use a larger network made of the first four blocks of VGG-11 combined with the original fully connected layers. The annotator model has been trained with the full set of images for 50 epochs with a learning rate of 0.001. The edge models have been pre-trained with the baseline dataset for 50 epochs with a learning rate of 0.001. Edge model retraining in the cloud takes place for 5 epochs with a learning rate of 0.001

5.3. Metrics

In order to test the classification performance of the edge model we use held-out datasets both from the Stream and the Baseline image groups. We use the following metrics:

- *Stream Accuracy*: Accuracy of the model evaluated with the held-out dataset from the set S of Stream images. This metric measures a model's ability to learn from new samples.
- *Baseline Accuracy*: Accuracy of the model evaluated with the held-out dataset from the set B of Baseline images. This metric shows the model's ability to maintain its effectiveness in classifying samples that follow the same distribution as the training samples (i.e., its ability to avoid catastrophic forgetting).

The edge model is evaluated at each time step using both metrics.

6. Results

The proposed *Dynamic Edge-Cloud* method is compared against the following baselines:

- *Static Edge*: A static, pre-trained model, which is only used for inference.
- *Driver-based Edge*: A model which is trained only with the driver actions as labels. The model is trained at the edge for every time step. Pre-training takes place only once in the beginning and is optional, we use a learning rate of 0.01 for driver-based edge training since we need fast adaptation to the new data in the online setting.

We evaluate the performance of the three approaches (proposed and the two baselines) in three different scenarios, which are discussed in the following sections.

Note that blue triangles indicate when cloud retraining has been triggered by a predefined temporal threshold and not due to the activation of the drift detector. Red stars indicate time-steps when cloud retraining was triggered by the drift detector.

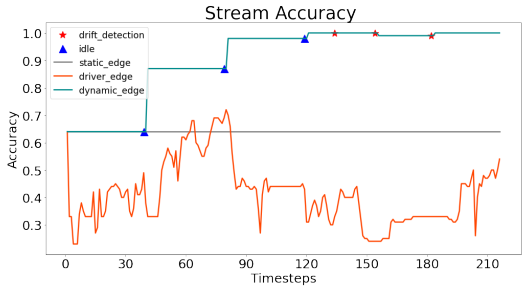
6.1. Perfect Driver - Pre-trained Model Test

This scenario assumes that the driver never makes a mistake and the model on the edge has already been pre-trained using the baseline dataset. It demonstrates how a driver, an edge model and the cloud infrastructure can interact with each other. The respective results in Figures 3a) and 3b) show that the proposed method quickly outperforms the two baselines, both in terms of the Stream and Baseline held-out datasets. Compared to the "Static Edge" model, the proposed method uses additional images for training as these become available. On the other hand, the "Driver-based Edge" uses the same data as our method, however it achieves lower performance due to catastrophic forgetting, since the stream data is non-i.i.d.

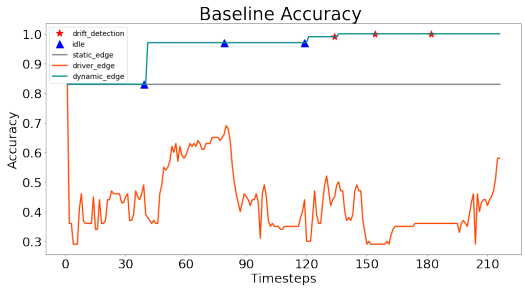
6.2. An occasionally incautious driver with a pre-trained model

The second scenario evaluates the ability of the proposed mechanism, even when it receives false feedback from the driver.

The driver may respond in the wrong manner with a predefined error ratio (i.e. 15% in this case) and the most common mistake being the disregard of a traffic sign. For this reason we have created a pseudo-random function where the driver assigns the label 0 in a traffic sign where an action is required (i.e. halt or decelerate). Figures 4a) and 4b) show that the proposed method is fault-tolerant even in the case of a incautious driver, thus providing

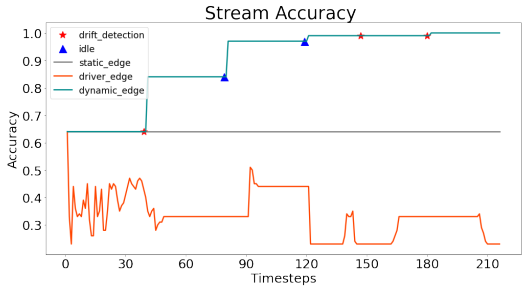


a)

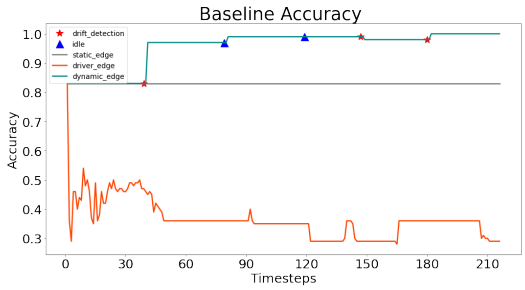


b)

Figure 3: a) Stream accuracy and b) Baseline Accuracy for the best case scenario, in which the driver always makes the right decision. with 1) an edge without further training 2) an edge model which trains with driver input and 3) cloud-edge combination.



a)



b)

Figure 4: a) Stream accuracy and b) Baseline Accuracy for "incautious driver" case scenario with 1) an edge without further training 2) an edge model which trains with driver input and 3) cloud-edge combination.

a reliable solution which can maintain and expand its ability to classify.

6.3. Train a model from scratch with a perfect driver

The last scenario focuses in training an edge model from scratch. In this occasion, we assume that the driver is a professional and that the pre-training phase has been omitted. The objective is to see the difference between training instantly on the edge while having only the driver responses versus the cloud-edge framework. Note that in the case of the Driver-based edge, we never use the baseline dataset, therefore there is no point of measuring Baseline Accuracy. Figure 5 illustrates the Stream Accuracy for the three methods. Despite the fact that the Driver-based edge will work instantly on improving the edge model, it can be quite unstable whereas the proposed method provides reliable classification when the edge decides to send the disagreement buffer in the cloud.

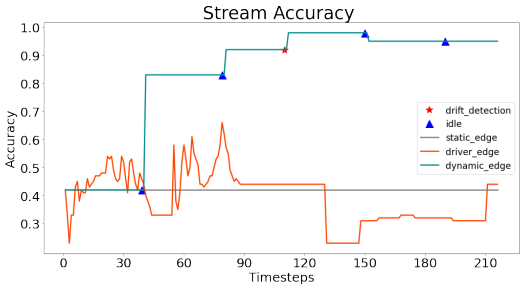


Figure 5: Stream accuracy for "training from scratch" case scenario with a) an edge without further training b) an edge model which trains with driver input and c) dynamic cloud-edge combination.

7. Conclusions

This work examined a framework for continual learning on traffic sign classification systems as a part of assisted/autonomous driving. Our suggestions use the cloud infrastructure in order to ensure that a suggestion model can expand its classification range while working under constrained hardware specifications. The results show that we can achieve high classification scores both on

terms of incorporating new knowledge and maintaining a baseline standard of classification accuracy.

Acknowledgments

This work is supported by the "TEACHING" project that has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement No 871385. The work reflects only the author's view and the EU Agency is not responsible for any use that may be made of the information it contains.

References

[1] M. D. Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. G. Slabaugh, T. Tuytelaars, Continual learning: A comparative study on how to defy forgetting in classification tasks, CoRR abs/1909.08383 (2019). URL: http://arxiv.org/abs/1909.08383. arXiv:1909.08383.

[2] C. Davalas, D. Michail, C. Diou, I. Varlamis, K. Tserpes, Computationally efficient rehearsal for online continual learning, in: Image Analysis and Processing - ICIAP 2022 - 21st International Conference, Lecce, Italy, May 23-27, 2022, Proceedings, Part III, volume 13233 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 39–49. URL: https://doi.org/10.1007/978-3-031-06433-3_4. doi:10.1007/978-3-031-06433-3_4.

[3] A. Cossu, D. Bacciu, A. Carta, C. Gallicchio, V. Lomonaco, Continual learning with echo state networks, CoRR abs/2105.07674 (2021). URL: https://arxiv.org/abs/2105.07674. arXiv:2105.07674.

[4] J. Zhang, W. Wang, C. Lu, J. Wang, A. K. S, Lightweight deep network for traffic sign classification, Annales des Telecommunications/Annals of Telecommunications 75 (2020) 369 – 379. doi:10.1007/s12243-019-00731-9.

[5] D. Tabernik, D. Skocaj, Deep learning for large-scale traffic-sign detection and recognition, CoRR abs/1904.00649 (2019). URL: http://arxiv.org/abs/1904.00649. arXiv:1904.00649.

[6] O. Ali, M. K. Ishak, Bringing intelligence to iot edge: Machine learning based smart city image classification using microsoft azure iot and custom vision, 2020.

[7] G. Widmer, M. Kubát, Learning in the presence of concept drift and hidden contexts, Machine Learning 23 (2004) 69–101.

[8] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang, Learning under concept drift: A review, IEEE Transactions on Knowledge and Data Engineering (2018) 1–1. URL: https://doi.org/10.1109%2Ftkde.2018.2876857. doi:10.1109/tkde.2018.2876857.

[9] G. J. Ross, N. M. Adams, D. K. Tasoulis, D. J. Hand, Exponentially weighted moving average charts for detecting concept drift, Pattern Recognition Letters 33 (2012) 191–198. URL: https://doi.org/10.1016%2Fj.patrec.2011.08.019. doi:10.1016/j.patrec.2011.08.019.

[10] R. Timofte, K. Zimmermann, L. Van Gool, Multi-view traffic sign detection, recognition, and 3d localisation, volume 25, 2009, pp. 1–8. doi:10.1109/WACV.2009.5403121.

[11] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, CoRR abs/1409.1556 (2014). URL: http://arxiv.org/abs/1409.1556.