Oslo: machine learning across multiple edges

Muhammad Balagamwala[1], Joshua Cha[6], Sina Chavoshi[6], Luis Miguel Hernanz[1], Jeff Oxenberg[1], Lin Nease[4], Raj Ramanujam[6], Tzach Segal[5], Amip Shah[2], Paul Turner[3], Rajesh Vijayarajan[1]

[1] HPE Business Enablement, Solutions, and Technology (BEST) [2] Hewlett Packard Labs

[3] HPE Pointnext [4] HPE Global Industry Solutions (GISA) [5] HPE Alliances [6]Google

{firstname.lastname} @hpe.com {joshuacha, rajrajamunajam, chavoshi} @google.com

Note: authors listed alphabetically by last name

Abstract

We introduce Oslo, a distributed edge-cloud solution for machine learning across multiple edges. As proof-of-concept, we demonstrate Oslo for an image classification problem where a (hypothetical) manufacturer seeks to validate that a robotic arm has picked up the correct type of bolt on an assembly line. We have recently engaged the first customer for Oslo (a medical imaging company); and are working with the HPE Pointnext and the Google alliances team on productization.

Problem statement

Customers are increasingly training machine learning (ML) and deep learning (DL) models in a central core and then deploying these models at the edge for real-time inference. This helps with reducing the latency associated with inference [1], but input data still needs to be moved to the central core for purposes of updating the model ("lifelong learning"). As a result, two potential benefits of the edge—reducing the costs associated with moving data; and being able to keep data private to each location—do not get realized. In addition, particularly in deployment scenarios involving multiple edges, it can be valuable to combine insights from each edge to create an updated model and to optimize the model for each edge. To do so, transfer learning[1] is emerging as a preferred approach—where only a few layers of the neural network requiring updating are retrained, so the (re)training overhead can be reduced. Lastly, we see many customers boot-strapping edge-to-core ML solutions through custom integration of different technologies—requiring significant NRE investment and long time-to-value. To address this painpoint, model deployment and retraining must be automated from end-to-end in an easily deployable fashion.

To summarize, a solution is needed which: (i) enables ML/DL inference on the edge (with model training in a centralized core); (ii) minimizes the data flow from edge to core (especially for model updating); (iii) supports transfer learning at scale; and (iv) provides an automated / integrated workflow across edge and core.

Our solution

To address the above needs, we have created "Oslo"—a hybrid edge-cloud solution for machine learning across multiple edges. While Oslo is architected to be generically applicable for any ML model, we discuss (and initially demonstrate) in the context of image classification. The exemplar use case we use is that of real-time object recognition, focusing on bolts. (A common problem in manufacturing is that a wrong bolt gets placed into a pre-sorted bin and picked up by an industrial robot. When this happens, the entire part that mates with the bolt typically needs to be discarded. A window of a few seconds exists where—if the type of bolt picked up by the robot can be identified, and an alert generated if the wrong type of bolt is picked up—then costly waste can be avoided.)

For our exemplar use case, we use a trained ML model that recognizes nine different types of bolts. Due to a local inventory shortage, one of the factories decides to use a slightly different part with a new type of bolt. In traditional workflows, the factory would have been required to identify the new type of bolt that is going to be introduced; images of the tenth bolt type would need to be created; a new model would be trained; and the model would then be manually triggered for roll-out. Using Oslo, by contrast, the customer would simply onboard the new part in an offline trial run; the Oslo platform would identify the new types of bolts as outliers; these would be labeled as a

[1] Traditionally, transfer learning is used for situations involving domain adaptation, while lifelong learning implies using the past (*n-1*) tasks to improve learning of the same task at the *n*-th timestep. In our case, because we have multiple edges which may be focused on the same single task or on tasks across domains, the solution must support both interchangeably.

new bolt category using the Oslo dashboard; and the rest of the process—training, updating, and deployment of the new model—would be automatically handled in the Oslo workflow. Figure 1 demonstrates how Oslo achieves this:

1) **Edge Ingest**: enables connections to the source of the data. (Fig. 1 assumes that the bolt images are first deposited into a local folder; in practice, these can also be streamed directly to the Edge Analysis module.)

2) **Edge Analysis**: where local inference happens. We currently use Tensorflow, a widely used ML platform that is integrated into Google Cloud Platform (GCP) with built-in capabilities for model serving, versioning, and transfer learning. For our use case, the output will be the % likelihood that an ingested image is a certain type of bolt.

3) **Edge Visualization**: a dashboard showing performance of the model over time. This is also where the user can customize parameters related to model inference, e.g. the desired confidence level associated with the classifier.

4) **Retraining Pipeline**: images that fall below the set confidence level are collected into a local folder on the edge. A watcher transfers any such outlier images to a GCP storage bucket. (Other images are never moved off the edge.)

5) **Global Model Training**: images from the storage bucket are fed to CloudML via a set of GCP services. CloudML outputs a new trained model, which is containerized and stored in the Google Container Registry.

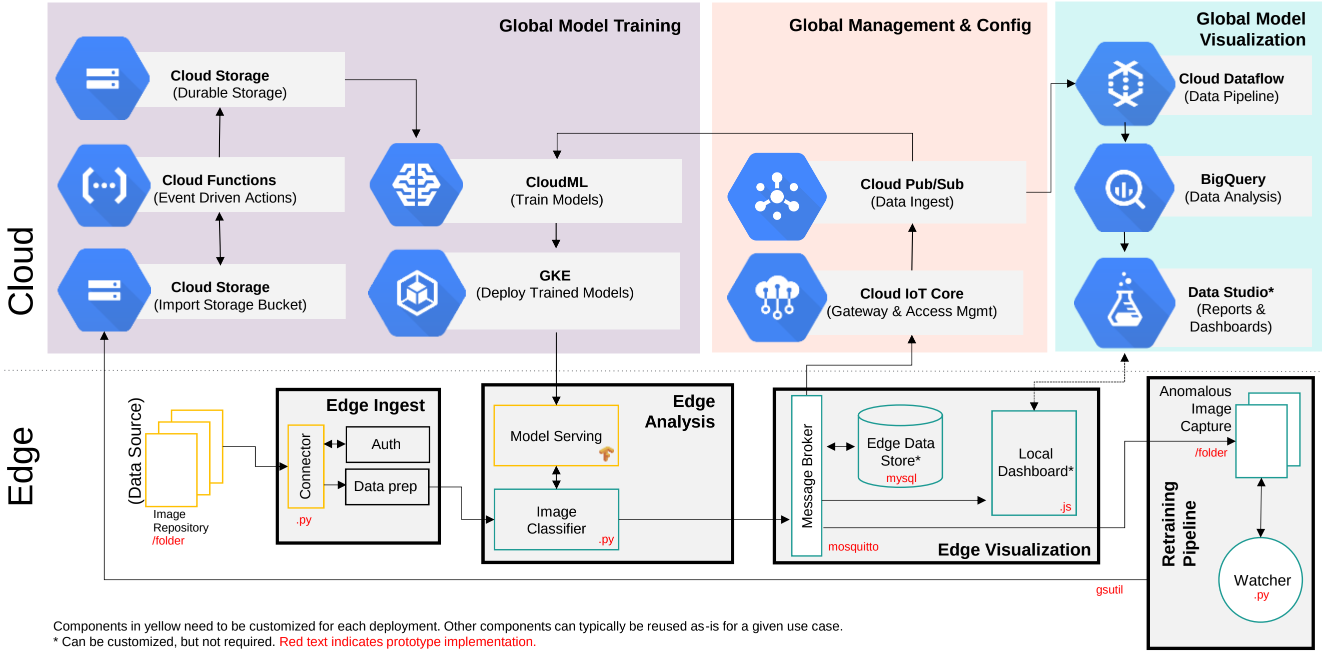
6) **Global Model Management and Configuration**: we leverage GCP's IoT Core service to facilitate device management and configuration, including enabling an edge to proactively request a new model from the core. We can also use Cloud IoT Core to send a message to all edge nodes to self-update once a new model is available.

7) **Global Model Visualization**: statistical data about model performance on each edge are streamed to a central dashboard, so that any underperforming edges (which may warrant a model upgrade) can be detected.

An important feature of the solution is its microservices-based architecture, which relies upon containers and cloud functions to deliver the components within each module. This allows for the implementation of any of the edge modules to be customized, without adversely affecting the larger solution. For example, a customer could choose to have different versions of the inference engine running in parallel on different edges.

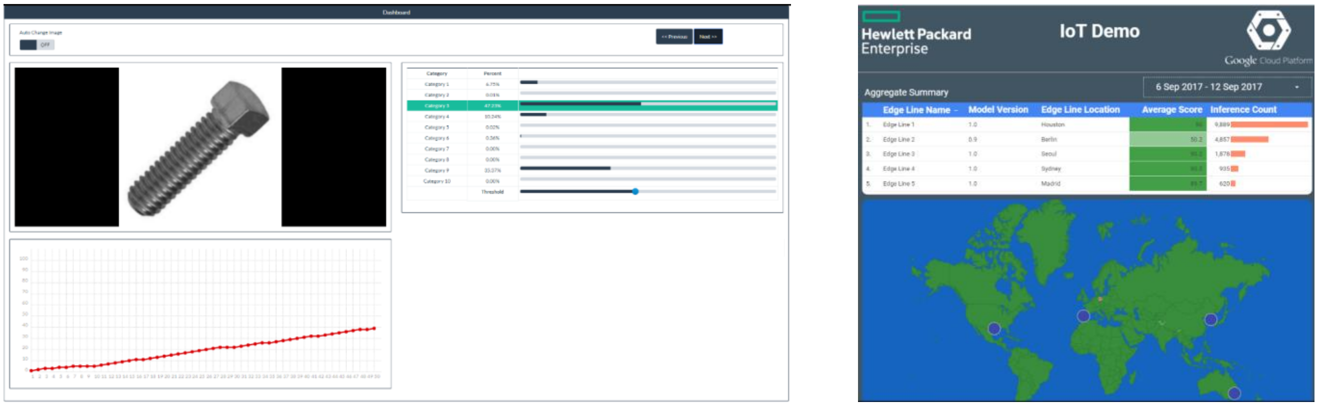
Evidence the solution works

We have developed a prototype solution for the above manufacturing use case. Figure 2 shows the dashboards on the edge and cloud respectively. The classifier and datasets were provided by Google and retraining / transfer



Components in yellow need to be customized for each deployment. Other components can typically be reused as-is for a given use case.
* Can be customized, but not required. Red text indicates prototype implementation.

Fig. 1: Solution Design. Oslo is comprised of two parts: a set of edge functions, implemented locally (e.g., on an HPE Edgeline server); and a set of cloud services (implemented using Google Cloud Platform). NOTE: although shown for only one edge, the above can be scaled for *n* edges.



Category	Percent
Category 1	6.75%
Category 2	0.01%
Category 3	47.23%
Category 4	10.24%
Category 5	0.02%
Category 6	0.36%
Category 7	0.00%
Category 8	0.00%
Category 9	35.37%
Category 10	0.00%
	Threshold

Aggregate Summary

	Edge Line Name	Model Version	Edge Line Location	Average Score	Inference Count
1.	Edge Line 1	1.0	Houston	[illegible]	9,889
2.	Edge Line 2	0.9	Berlin	50.2	4,857
3.	Edge Line 3	1.0	Seoul	[illegible]	1,878
4.	Edge Line 4	1.0	Sydney	[illegible]	935
5.	Edge Line 5	1.0	Madrid	[illegible]	620

(a) **(b)**

Fig. 2: Dashboards for proof-of-concept use case (a) on one Edge. We see the relative confidence level that the given bolt falls into each of 10 different categories, as well as the model performance over time. **; (b) globally on GCP**. We see how each model has been performing relative to each other.

learning was achieved using Inception [2]. Importantly, the edge functions are hosted on an EL1000 in Palo Alto that sits behind the HPE firewall, while the global (cloud) functions are on a private instance of GCP—the solution successfully works through practical issues related to authentication, port binding, container management, etc.

Competitive approaches

Google and Nutanix recently announced a collaboration [3] where Tensorflow models trained on GCP can be deployed on Nutanix infrastructure at the edge, but this solution does not have the automated end-to-end workflow or model management that we provide. The closest alternative to Oslo would be Amazon Greengrass, which supports an edge deployment of AWS Lambda functions, but does not currently integrate with Amazon ML. The idea of sending (just) outliers back to the core to minimize data flow was previously proposed in HPE's Distance Deep Learning (DDL) project [4]. With multiple edges, however, the DDL model at one edge can drift from that deployed at another edge—and merging outliers from all edges to create a global model would require longer periods for model (re)training. Our solution enhances DDL by introducing native support for transfer learning.

Current status

We currently have a functioning prototype, and recently started introducing the solution to customers under NDA. We are now qualifying a pipeline of healthcare, logistics, and oil & gas companies; and have just kicked off a proof-of-concept with our first customer (a medical imaging company). In parallel, we are also working with HPE Pointnext and Google (as a DL/ML partner) on defining a joint offering based on Oslo.

Next steps

We plan to further test the platform with alternative datasets, different models beyond image classification, as well as other ML/DL platforms (e.g., Caffe or Pytorch). In the longer term, we will also explore opportunities for Oslo within HPE's broader AI efforts [5, 6] as well as HPE's Edge-as-a-Service [7] and Universal IoT Platform offerings.

References

[1] P. Jain, et al., "Real-time Video Analytics on the Edge," HPE TechCon 2017.

[2] J. Shlens, "Train your own image classifier with Inception in TensorFlow," https://research.googleblog.com/2016/03/train-your-own-image-classifier-with.html (March 9, 2016).

[3] www.nutanix.com/press-releases/2017/06/28/nutanix-teams-google-cloud-fuse-cloud-environments-enterprise-apps/

[4] Daniel Wu, et al., "Distance Deep Learning," HPE Discover 2017 (Las Vegas, NV).

[5] N. Vassilieva, et al., "Deep Learning Cookbook," HPE Discover 2017 (Las Vegas, NV).

[6] Gunalan P.V., et al., "Dataflow and ML Model Management for IoT Intelligent Edge," Abstract to HPE TechCon 2018.

[7] A. Shah, et al., "Towards Edge Infrastructure as a Service," Abstract to HPE TechCon 2018.