Shadow Puppets: Cloud-level Accurate AI Inference at the Speed and Economy of Edge

Srikumar Venugopal
IBM Research – Ireland

Michele Gazzetti
IBM Research – Ireland

Yiannis Gkoufas
IBM Research – Ireland

Kostas Katrinis
IBM Research – Ireland

Abstract

Extracting value from insights on unstructured data on the Internet of Things and Humans is a major trend in capitalizing on digitization. To date, the design space for doing AI inference on the edge has been highly binary: either consuming cloud-based inference services through edge APIs or running full-fledged deep models on edge devices. In this paper, we break this design space duality by proposing the Semantic Cache, an approach that blends best-of-breed features of the extreme ends of the current design space. Early evaluation results on a first prototype implementation of our semantic cache service on object classification tasks shows tremendous inference latency reduction, when compared to cloud-only inference, and high potential in scoring adequate accuracy for a plurality of AI use-cases.

1 Introduction

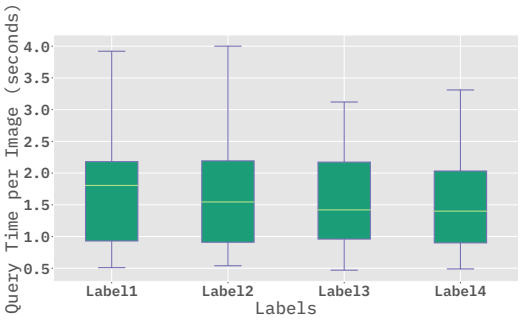
Trends show that extracting value from edge data by moving sensor data to the cloud is not an economically viable strategy, as it increases the cost of insights and also leads to diminished returns due to prohibitive time-to-insight (latency). The problem is more pronounced in the realm of extracting insights from unstructured data - typically by applying AI techniques - due to increased data density and an increasing number of sensor devices capturing video and audio (e.g. smartphones). The latter trend exercises extreme pressure to metropolitan networks in their capacity to continuously stream large volumes of data upstreams towards cloud datacenters. But even if new trends (e.g. 5G networks) manage to close this gap, the cost of this endeavour puts the analytics on the edge investments at risk.

Edge Computing [16] carries the promise of inversing this disporptional situation, namely by moving general purpose compute capability closer to the data. Particularly applying to unstructured data, this paper presents first a design space exploration (Section 2) for AI inference on the edge. Through rigorous experimentation on cloud providers, we first argue on a data-backed basis on the limitations of cloud-based AI for edge use cases. We then turn to the other end of the design space, namely by evaluating the straightforward approach of deploying deep models on (powerful) edge devices.

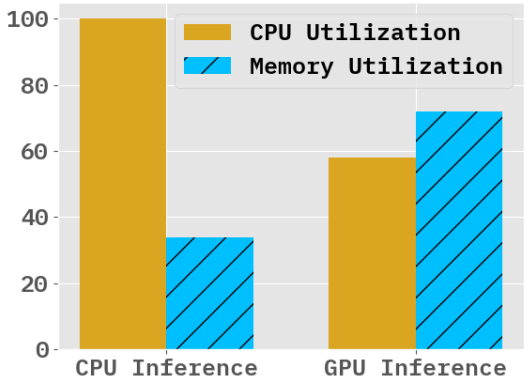
In this paper, we propose to break the to-date binary design space of AI inference on the edge, particularly through a best-of-both-worlds system and approach termed *Semantic Cache*. As its name conveys, our approach employs a commonly used technique (caching) to mask the latency caused by using cloud services to perform inference on unstructured data; it heavily economizes on cost of bandwidth and strain put on metropolitan networks and cloud gateway by significantly reducing the volume of unstructured data that needs to be sent back to the cloud. We present the design and techniques used to transition caching to the semantic domain (Section 3). We have implemented a mature prototype of the semantic cache on high-end edge devices and used this prototype to evaluate the efficacy of the approach on visual object classification tasks using public datasets (Section 4). Our early results manifest the competitiveness of the approach: our semantic cache achieves signifant trimming of inference latency, when compared to cloud-based inference, at a controlled accuracy cost.

2 Binary Design Space Exploration

In this section, we discuss an experimental evaluation of the two far ends of the design space for completing object classification tasks on the edge. In the first experiment, we evaluate the feasibility of completing object classification tasks on edge devices by having them fetch inference results from production cloud services. Moving to the other far end of the design space, we analyse the impact of executng full deep models for inference on resource usage of edge devices. We use our findings to



(a)



(b)

Figure 1: (a) Measured latency (in seconds) on an edge device in having images streamed and having objects appearing on these images recognized in the cloud. Results are grouped per actual image label and have been aggregated across all four cloud providers tested. Along with minimum and maximum latency, 25/75 percentile (low and top end of the box plot) and median values (line within box) are shown; (b) Resource utilization incurred by executing the Resnet-152 CNN on an NVIDIA TK1 SoC. Both CPU-only and GPU-accelerated inference results are shown.

motivate the case for novel approaches that combine the best features from these binary extremities.

2.1 AI Inference in the Cloud

We instantiated a custom object recognition client running on a COTS (commercial-off-the-shelf) Linux-based edge device (NVIDIA Jetson), whereby the client fetches images from the local flash storage sequentially and consumes a cloud-based visual recognition service to retrieve the top-5 probable labels corresponding to the single object shown in each image. We conducted four distinct trials of this experiment, one for each of four out of the top-10 cloud computing vendors worldwide [5].

In each trial, the client consumes images from a subset of 50 images randomly selected from the Large Scale Visual Recognition Challenge 2012 dataset (a.k.a. Imagenet) [14] and uses the cloud inference API offered by the cloud provider to retrieve object labels. In all subsets, each image depicts an object belonging to one out of four object types, i.e. the set of the ground truth labels equals to four. For each image recognition request, the client measures the incurred recognition latency. Throughout the experiment trials, the edge device resided in Ireland (Europe), while the cloud providers' service endpoints have been selected to all reside in datacenters located in West North Central U.S. states.

We have combined the latency results obtained from the above experiment and grouped them per ground-truth label (equivalently object type) across all four cloud providers, as shown in Figure 1a. While there is no correlation between the labels and the latency range of inference, it is apparent in this figure that **the cloud-based inference incurs high overhead and an almost 10x variance from minimum to maximum**. Specifically, **the average inference latency has been found in the order of 1.5 seconds, while the high percentile latency exceeds the 2 seconds front**. In the best case, inference rate could not exceed 2 frames per second.

For use-cases posing near real-time response requirements such as e.g. augmented/virtual reality, autonomous vehicles and robotics applications, incurred cloud inference latencies seem prohibitive. But even beyond the range of applications with stringent real-time requirements, capping inference solutions at 2 frames per second can be detrimental to scenes exhibiting objects with low dwell time. For instance, with 60% of road signs as captured from a moving vehicle exhibiting a dwell time lower than 2 seconds [2], cloud-based road sign visual recognition seems currently far from achieving autonomous driving safety compliance regulations.

A technically viable alternative to achieve higher inference rates would be for cloud providers to offer pipelined services (e.g. by having a frontend cloud service dealing incoming frames to multiple instances of the internal inference service), only to add to the operational cost of the solution. A major contribution of this paper is in showing that there are more cost-effective ways for cloud service providers to arrive to the same desirable result through cloud-assisted edge-based inference.

2.2 AI Inference on the Edge

In this experiment, we extensively evaluated key performance indicators of running cutting-edge deep networks for object classification, using the NVIDIA Tegra TK1 evaluation board as the edge device and a subset of the test image set from [14] as input. The TK1 board carries a quad-core ARMv7 (A15) processor, along with 192 Kepler CUDA cores. Due to space limitations, we only present results - shown in Figure 1b - for one of the tested convolutional neural networks (RESNET-152 on Caffe). In one set of our experiments, we disable the use of the CUDA cores during inference, effectively to assess resource utilization of the deep network on inexpensive general-purpose edge SoCs. As shown in Figure 1b, **edge-based object recognition is depleting CPU resources, effectivelly preventing the execution of any further services or applications on the SoC. When GPU-based inference is turned on, the typically constrained memory of the edge device was found to be highly utilized by the deep network executable**.

Overall and as also experimentally confirmed in the literature [1], there is a hyperbolic relationship between accuracy and resource utilization, making accuracy improvements highly expensive relative to the amount of resources found on conventional or even high-end edge devices. Recent approaches (further discussed in Section 5) have tried to reduce the footprint of such models on constrained devices, albeit without removing practical limitations such as constant need for model (re-)training. Common to the footprint of the model (i.e. compressed or not), a shortcoming stemming from this approach is the inability to execute model ensembles on embedded edge devices. While difficult to confirm due to confidentiality limitations, this is highly likely to be case for commercially available cloud-based object recognition services, judging from their ability to achieve high levels of accuracy in generalized input setting. As we will show in the following, our approach is able to benefit from the superior characteristics of cloud-based inference services, while being directly competitive to edge-only inference in terms of resource utilization and performance.

3 Semantic Cache Service Design

Figure 2 illustrates the architecture of our semantic cache service approach for inference on the edge. The cache interface provides the entrypoint for the client to interact with our edge service. The client submits an image or video frame for inference to the cache service via the cache interface. The interface backend passes the image to the encoder, which in turn extracts key image features and thus converts the image into a standard representation, specifically a bit vector resembling a compressed

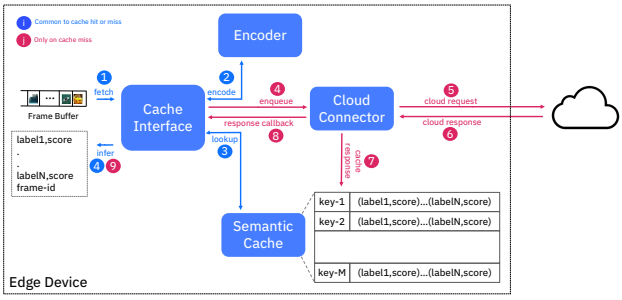


Figure 2: Block diagram of the Semantic Cache service.

encoding of its features.

The encoding is first used to search through the cache for a match. If there is a cache miss, then the image is sent to the cloud service for inference. The inference result from the cloud service is generally a set of labels and their confidence scores. The result of the cloud query is stored in the cache indexed by the encoding as the key, and finally returned to the client. Subsequent hits against the same feature representation will result in the inference results being returned from the cache, rather than an expensive query being sent to the cloud.

A key design decision in the semantic cache service is the feature encoding approach. Temporal variance in the imaging data arriving at the edge devices implies that the chances of an input exactly matching the keys stored in the cache are extremely minimal. Hence, the requirements for the feature extraction and representation are to provide a signature for the image or video frame such that approximate matching can be performed to obtain results from the cache (i.e. achieve a relatively competitive hit rate at high accuracy and recall levels). Cachier [4] has employed a standard feature extraction and classification pipeline (e.g. SIFT combined with Locality Sensitive Hashing) for this purpose. However, image comparison on the basis of feature vectors that encode pixel information, may not reflect the similarity as perceived by humans.

Our approach is to use a small-footprint neural network to generate hash codes that represent the images or video frame data. The ability of neural networks to learn the feature representation of an input space has been key to their success in dealing with image and video generated by neural networks are a representation of the image features in the "semantic" space, i.e., images belonging to a similar class or context have encodings that are close to each other by a chosen distance measure [15].

Krishevsky and Hinton [8] demonstrated how to use autoencoders, a type of neural networks, in this manner for image retrieval. More recently, Lin, et. al [10] have used deep convolutional neural networks to learn a low-dimensional binary representation of an image. In

this scheme, a hidden layer is introduced before the final classification output layer of a CNN during training. This layer learns a binary representation of the image by applying a sigmoid function to the activations obtained from the preceding layer. This hash code can be used for comparing different images over Hamming distance.

While our cache architecture is independent of the encoding method, we have used the above scheme in our initial proof of concept implementation. The Hamming distance threshold is an input parameter specified by the user as part of an inference request. Our retrieval technique prioritises on direct hits, then uses the user-specified distance threshold to collect cache hits results within threshold. Subsequently, cached labels passing the hit criteria are coalesced and sorted by confidence level. These are returned to the user as the result of an inference request to the semantic cache service.

4 Prototype and Evaluation

We have matured a complete proof of concept implementation of the semantic cache, following an "as a service" design approach and packaging, exposing the service to edge device/gateway users via a REST interface. The service implementation employs purpose optimized module implementations for the heavy processing stages of the data path (encoder and cache lookups). We have successfully tested our prototype implementation on commercial edge-grade devices (Raspberry Pi 3 and NVIDIA Jetson TK1/Tx1 boards). Particularly for the aforementioned heavy processing stages and with the ultimate aid to offer extreme low inference latency, our implementations takes full advantage of acceleration features available on edge devices (specifically Kepler CUDA cores on the Jetson devices). We are continuing to take advantage of such opportunities on other more inexpensive devices (e.g. use vector instructions on ARMv7/8 devices). Our cloud connector module supports connectivity to production-grade cloud inference services, with the current implementation consuming IBM Watson Visual Recognition cloud service [6]. Support for further cloud services is a trivial step given our modular service design.

We have conducted a first evaluation of our semantic cache approach, using the the prototype implementation outlined above. We evaluated the semantic cache on completing object classification tasks on single object videos replayed as inference requests to the service. We used the Youtube-Objects video dataset [12] as input to our evaluation, including the use of dataset-provided annotated ground truth off the service data-path to evaluate service accuracy and recall. In this first evaluation, we deployed the well-known Darknet [13] framework on a server-class server (within 1 sec latency from the edge

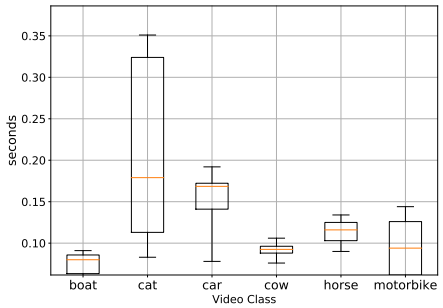


Figure 3: User perceived per-frame latency (in seconds), aggregated on a per object/image type/class basis.

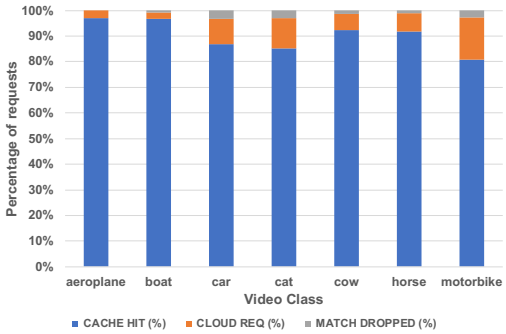


Figure 4: Cache efficacy results in infering similarity on visual input, assumming a hot cache.

device) to function as the backend cloud service for answering the image inference queries. We used this bespoke cloud service in order to gain full control of the various variables, such as latency and accuracy, involved in answering image inference queries.

We focus our initial evaluation on two key performance indicators, in light of showcasing the competitiveness of the approach vs. the two binary design space approaches discussed in Section 2: a) user experienced image inference latency averaged over a set of images, and b) precision and recall of the inference results returns to the user by the semantic cache service.

Figure 3 presents the per video frame inference latency, as perceived by the user posting inference requests to the semantic cache service. With the exception of a single outlier class ("cat"), the results show a remarkable 5x to 10x latency improvement, when compared to the cloud-only inference latency (cf. Figure 1a). Figure 4 delves deeper into the efficacy of the cache in terms of hits in the semantic domain. The results are inline with the latency results per class shown in the previous figures, as higher miss rate implies more frequent visits to the costly (latency-wise) cloud inference service. When further inspecting the "cat" class input, we have found

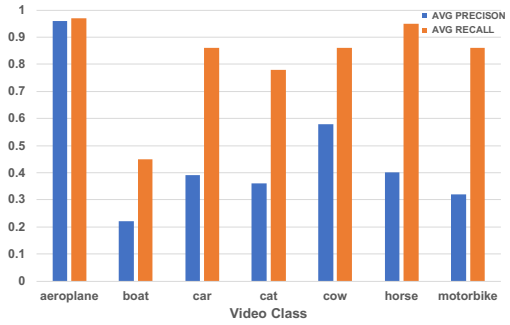


Figure 5: Precision and Recall Results

that in addition to the higher misses, the bespoke cloud model employed incurred higher latency in returning results to the edge device. We believe this to be an experimental artifact and not associated to the efficacy of the service in unveiling similarity in the semantic domain.

It is straightforward that inference performance is per se of low value, if it does not come together with adequeate inference accuracy. Figure 5 shows accuracy and recall results on the annotated dataset used. Overall, recall of the service is relatively high (almost comparable to the cloud service alone). On the other hand, we have observed an initial not insignificant penalty in precision. We attribute this to our ability of our initial encoder to distinguish correctly between classes that are characterized by high feature similarity. Controlling the trade-off between increased precision vs. resource footprint in our encoder module is part of our on-going work.

5 Related Work

Researchers have sought to produce hardware optimisations to execute DNNs in resource-constrained contexts. Sze, et. al [17] provide a comprehensive survey of this approach, examples of which are Eyeriss [3] and big/little DNN [11]. However, the development cycles for hardware also tend to be slower compared to the agile nature of software development that is being used to continuously improve neural network architectures for different applications. Other researchers have been looking into reducing the size of DNNs so that they can fit into constrained edge resources. Neurosurgeon [7] performs a trade-off between communication latency of querying the main DNN running in the cloud with the reduction in accuracy of querying a smaller DNN running on the edge device, based on the latency budget available for a single query and the network bandwidth between the edge and the cloud. In this case, there is a requirement for a smaller DNN that is customised for the same application as the cloud DNN. Deepx [9] takes a DNN and optimises it for performance at runtime on resource-constrained devices. This is only possible for some neural networks. Larger DNNs simply may not be able to fit into the available memory on edge device.

There have been a few publications on caching inference results on the edge. Glimpse [2] is a system for continuous real-time object recognition and tracking for camera-equipped mobile edge devices. Glimpse selectively offloads key (”trigger”) frames of the camera video feed to a remote server. The remote server facilitates feature extraction, object labelling and bounding box demarcation on each trigger frame. In parallel, Glimpse employs an ”active cache” of frames, whereby at any instance of time, the cache contains a selected set of the frames that occured between two consecutive trigger frames. Once the set of features for a given trigger frame is returned by the remote server, the frames in the cache are input to a feature tracking algorithm to provide for more gradual tracking of identified objects and thus to increase the precision/recall of overall object tracking. Glimpse employs caching of video frame to speed up the accuracy of tracking and is not focused on the obtaining inference results. Cachier [4] is a system for caching inference results on edge servers for image recognition applications. Cachier extracts features from an image that are then classifed using pre-trained models based on Locality-Sensitive Hashing or SVMs. The cache then returns the object whose features are the closest match to the input. The key disadvantage of Cachiers design is the number of features that are retained per image which increases the latency of edge lookup.

6 Conclusions

In this paper, we have experimentally evaluated the state of the art design space for AI inference on unstructured data on the edge. Extending beyond state of the art, we have introduced a novel approach termed Semantic Cache, which blends best-of-breed features of the extreme ends of the current design space and offers them as field controlled trade-offs that can be custom selected to match use-case KPIs at will (latency/cost vs. accuracy). Early evaluation of a first prototype implementation of our approach has showcase the potential of the approach in breaking the binary design space.

We are continuing to mature the initial approach, prioritizing on experimenting with alternative encoding techniques for improved precision, while also extending the evaluation to further datasets and AI tasks. We are also extending the Semantic Cache to deal with individual objects in a video frame by exploiting techniques in object detection to produce the image encoding [13].

References

[1] CANZIANI, A., CULURCIELLO, E., AND PASZKE, A. Evaluation of neural network architectures for embedded systems. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)* (May 2017), pp. 1–4.

[2] CHEN, T. Y.-H., RAVINDRANATH, L., DENG, S., BAHL, P., AND BALAKRISHNAN, H. Glimpse: Continuous, Real-Time Object Recognition on Mobile Devices. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems* (New York, NY, USA, 2015), SenSys '15, ACM, pp. 155–168.

[3] CHEN, Y. H., KRISHNA, T., EMER, J. S., AND SZE, V. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. *IEEE Journal of Solid-State Circuits 52*, 1 (Jan. 2017), 127–138.

[4] DROLIA, U., GUO, K., TAN, J., GANDHI, R., AND NARASIMHAN, P. Cachier: Edge-Caching for Recognition Applications. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (June 2017), pp. 276–286.

[5] EVANS, B. The cloud wars top 10: The world's most-powerful cloud-computing vendors. https://www.forbes.com/sites/bobevans1/2017/06/05/meet-the-cloud-wars-top-10-the-worlds-most-powerful-cloud-computing-vendors/, 2017.

[6] IBM. Watson visual recognition service. https://www.ibm.com/watson/services/visual-recognition/, 2018.

[7] KANG, Y., HAUSWALD, J., GAO, C., ROVINSKI, A., MUDGE, T., MARS, J., AND TANG, L. Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems* (New York, NY, USA, 2017), ASPLOS '17, ACM, pp. 615–629.

[8] KRIZHEVSKY, A., AND HINTON, G. Using Very Deep Autoencoders for Content-Based Image Retrieval. Ciaco. OCLC: 838104292.

[9] LANE, N. D., BHATTACHARYA, S., GEORGIEV, P., FORLIVESI, C., JIAO, L., QENDRO, L., AND KAWSAR, F. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *Information Processing in Sensor Networks (IPSN), 2016 15th ACM/IEEE International Conference on* (2016), IEEE, pp. 1–12.

[10] LIN, K., LU, J., CHEN, C.-S., AND ZHOU, J. Learning compact binary descriptors with unsupervised deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 1183–1192.

[11] PARK, E., KIM, D., KIM, S., KIM, Y. D., KIM, G., YOON, S., AND YOO, S. Big/little deep neural network for ultra low power inference. In *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)* (Oct. 2015), pp. 124–132.

[12] PREST, A., LEISTNER, C., CIVERA, J., SCHMID, C., AND FERRARI, V. Learning object class detectors from weakly annotated video. In *2012 IEEE Conference on Computer Vision and Pattern Recognition* (June 2012), pp. 3282–3289.

[13] REDMON, J., DIVVALA, S., GIRSHICK, R., AND FARHADI, A. You Only Look Once: Unified, Real-Time Object Detection. IEEE, pp. 779–788.

[14] RUSSAKOVSKY, O., DENG, J., SU, H., KRAUSE, J., SATHEESH, S., MA, S., HUANG, Z., KARPATHY, A., KHOSLA, A., BERNSTEIN, M., BERG, A. C., AND FEI-FEI, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV) 115*, 3 (2015), 211–252.

[15] SALAKHUTDINOV, R., AND HINTON, G. Semantic hashing. *International Journal of Approximate Reasoning 50*, 7 (July 2009), 969–978.

[16] SATYANARAYANAN, M. The Emergence of Edge Computing. *IEEE Computer 50*, 1 (Jan. 2017), 30–39.

[17] SZE, V., CHEN, Y. H., YANG, T. J., AND EMER, J. S. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE 105*, 12 (Dec. 2017), 2295–2329.