Declarative Data Serving: The Future of Machine Learning Inference on the Edge

Ted Shaowang
University of Chicago
swjz@uchicago.edu

Nilesh Jain
Intel Corporation
nilesh.jain@intel.com

Dennis D. Matthews
Intel Corporation
dennis.d.matthews@intel.com

Sanjay Krishnan
University of Chicago
skr@uchicago.edu

ABSTRACT

Recent advances in computer architecture and networking have ushered in a new age of edge computing, where computation is placed close to the point of data collection to facilitate low-latency decision making. As the complexity of such deployments grow into networks of interconnected edge devices, getting the necessary data to be in "the right place at the right time" can become a challenge. We envision a future of edge analytics where data flows between edge nodes are declaratively configured through high-level constraints. Using machine learning model-serving as a prototypical task, we illustrate how the heterogeneity and specialization of edge devices can lead to complex, task-specific communication patterns even in relatively simple situations. Without a declarative framework, managing this complexity will be challenging for developers and will lead to brittle systems. We conclude with a research vision for database community that brings our perspective to the emergent area of edge computing.

PVLDB Reference Format:
Ted Shaowang, Nilesh Jain, Dennis D. Matthews, and Sanjay Krishnan. Declarative Data Serving: The Future of Machine Learning Inference on the Edge. PVLDB, 14(11): 2555 - 2562, 2021.
doi:10.14778/3476249.3476302

1 INTRODUCTION

Model-serving systems are a crucial part of any modern machine learning deployment. These systems interface trained machine learning models (e.g., a neural network or an SVM) to software clients who can use those predictions (e.g., a fraud detection framework). The first iteration of these systems, including Clipper [10], TensorFlow Serving [46], and InferLine[9], were designed as RESTful cloud services. As the uses for machine learning have evolved towards increasingly latency and communication -sensitive applications, such as in control systems, industrial monitoring, and mobile applications, there has been a steady trend towards *moving model-serving to resources closer to the point of data collection.* We collectively call these computation resources "the edge".

Proceedings of the VLDB Endowment, Vol. 14, No. 11 ISSN 2150-8097.
doi:10.14778/3476249.3476302

The primary focus of recent research has been on reduced-size models that can efficiently be deployed on lower-powered devices [22, 24, 40, 63]. In our opinion, simply reducing computational footprint of each prediction served is an incomplete solution. As the database community learned with sensor networks [6, 17, 20, 64], there are significant data movement challenges in computing on decentralized data streams. For example, computing aggregate statistics over data from multiple different sensors requires smart communication strategies, such as aggregation trees, to minimize the network load while ensuring all required data is at "the right place at the right time" [13, 38]. Similar data movement problems resurface in many edge model-serving scenarios where data from multiple sources need to be combined for a prediction. However, to the best of our knowledge, no edge model-serving system orchestrates such data movement and they all rely on the user to ensure that featurized data is at "the right place at the right time" while reasoning about latency constraints.

As a concrete example, we highlight an example deployment of the DeepLens video analytics system [31] in a mock internet-of-things kitchen at the University of Chicago. Household activity recognition, where one uses information from smart devices in a household to determine what an occupant is currently doing, is an important task with applications in senior care and security. Accurate recognition in an unstructured household environment is quite complex and often requires aggregating multiple sources of information such as video, audio, and side-channel information (like network traffic) from IoT devices. As a part of the project, the team explored a machine learning model that could predict ongoing activities in real-time as a function of features of multiple video streams, audio streams, and network traffic captures from the IoT devices. There is a complex interplay between the design of the machine learning model, the necessary data flows needed to support that model, and the available resources in the edge network. For example, one could build a neural network whose predictions are based on a simple concatenation of features from all three sources, which would require aggregating the disparate data streams on a single node capable of running neural network inference. Or, one could imagine first using the volume from the audio signal to determine whether any activity is occurring, which then triggers more expensive video processing if necessary. Beyond these two choices, there are many more prediction architectures one could envision each of which has its own accuracy, communication, and computation trade-offs.

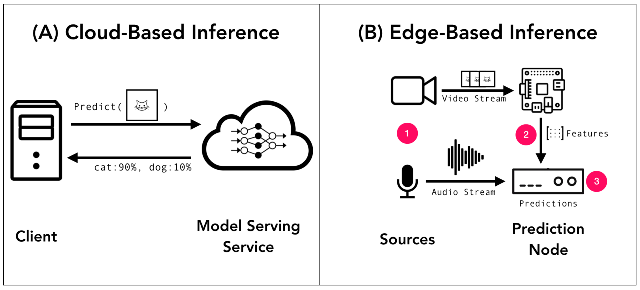

Figure 1: (A) A cloud-based model serving system. The client sends a prediction result, which is an image of a cat, and receives a prediction result from a model served in the cloud. (B) An edge-based model serving system. Video and audio data are continuously streamed to a server for activity recognition. The video data is first preprocessed by an embedded Intel Movidius VPU to calculate visual features.

Beyond this specific example, there are a number of common themes that we have observed in edge-based inference problems (Figure 1). First, data arrive in a streaming fashion (Figure 1B.1) supporting continuous, synchronous model predictions. This workflow differs from those in existing model-serving systems that assume asynchronous communication with a RESTful API. Next, edge resources, while increasingly capable, are often highly specialized for power, security, or other considerations. This specialization means that data often have to be moved between nodes, hereafter called "intra-edge data flows" (IeFs), for processing (Figure 1B.2). Finally, edge systems often interact or actuate the real world, and thus, predictions have to be materialized on specific nodes in the network (Figure 1B.3). This property leads to further IeFs between nodes in a network.

Here, we believe that the database community can contribute a new perspective that re-focuses the model serving systems towards optimizing IeFs. IeFs, when hand-written, can be brittle and lack the adaptivity to account for dynamic data arrival rates, resource availability, or contention. The database community has solved a number of these problems in the context of sensor networks, and in particular, promoted declarative programming to abstract away low-level communication decisions from the analyst writing the query. Ideally, IeFs should be managed in a similar way, where the user defines a high-level prediction workflow and locality constraints, and an automatic optimizer makes model placement and data routing decisions. This "data-first" perspective contrasts with the computation-first vision of current edge machine learning proposals that focus on making the actual machine learning inference tasks more efficient.

This paper describes an envisioned edge-based model serving system, called EdgeServe, that not only manages a machine learning inference service but also orchestrates data movement between nodes on an edge network. Practically, this means supporting the following features not considered in current cloud-based model serving systems:

(1) *Heterogeneous and disaggregated edge resources.* The system should automatically reason about a network of heterogeneous and disaggregated resources and automatically make cost-based judgments about model placement and data movement.
(2) *Complex and conditional prediction architectures.* The system should support complex task graphs where models can consume the prediction outputs from other models or can be triggered by those outputs.
(3) *Failures.* The system should robustly adapt to resource unavailability.

We believe such a system will create new opportunities for machine learning research on edge networks, including designing models that can be spatially decoupled (spread across multiple nodes with a slow network between them), models that are fault-tolerant (can issue predictions even if some data sources are lost), and AutoML techniques to search through complex axes between accuracy, communication and computation.

2 SURVEY OF CURRENT TECHNOLOGY

It is easy to dismiss EdgeServe as simply a reinvention of the sensor networks research of the 2000s [17, 20]. However, the very nature of the edge has changed in recent years due to computer architecture, networking, and infrastructure trends that substantially changed the assumptions that underpin such systems.

2.1 Hardware and Workload Trends

Trend 1. Specialized Edge Hardware. Recent advances in computer architecture research have resulted in highly-specialized edge devices in power-efficient form factors including edge visual processing systems for fast multimedia processing [25, 44], TPU systems for machine learning serving [19], programmable switches for fast packet analysis, and a variety of 5G-capable devices for high-bandwidth mobile applications [30]. In short, the edge is no longer "underpowered" as many in the 2000s assumed [32]. Our mental model for the future edge is not one that is under-powered, but rather one that is highly disaggregated with a complex network of heterogeneous computing and data collection nodes.

Trend 2. Growing "Public Edge" Infrastructure. Recent IT trends are blurring the line between content distribution networks (CDNs) and edge computing. Various commercial offerings allow users to not only geo-distribute data, but also computation [49]. ISPs are further expanding offers to co-locate computation at or near telecommunication base stations. In short, over the next decade, there will be an entire ecosystem of infrastructure-as-a-service that fills the void between edge and cloud. This ecosystem will serve to make edge computing less of an all-or-nothing proposition, and more of a user-defined trade-off of reliability, latency, and cost. More degrees of freedom in the design of such networks necessitates smarter tools to take advantage of them.

Trend 3. Prevalence of Smart Home Devices. Per research firm Statista, there will be nearly 2.7 billion smart home devices installed in the United States by 2023 [55]. The huge amount and variety of sensitive data generated by such devices will introduce new workloads for the entire data pipeline, from data collection to decision making. Current cloud computing paradigm is not efficient enough for such workloads, as it would be expensive, both in bandwidth and latency, to transfer a large amount of raw data to the cloud, let alone privacy and security concerns. Edge computing will fill the gap to shorten the distance between data and compute, and leave only featurized anonymous data to the cloud for offline analysis.

Trend 4. From Analytics to Decision Making. Workloads in edge networks have changed as well. In the sensor network work of the 2000s, the primary "real-time" workload was continuously updated dashboards and reports to summarize data for a human analyst [39]. Since then, the growing maturity of AI has made us more comfortable with automated decisions made fully by software systems. This change has led to a new era of latency-sensitive applications, in real-time control, robotics, security, and autonomous driving, as human action is no longer a decision bottleneck.

These advances in statistical machine learning will drive the workloads for future edge systems. As such, we will focus on model serving as a prototypical workload. There have been several recent model-serving frameworks including Clipper [10], TensorFlow Serving [46], and InferLine[9] aimed at cloud systems. Recently, there has been more interest in model-serving at the edge [11]. If the data needed to construct features resides on multiple nodes, it is the user's responsibility to bring that data together. Thus, in a disaggregated environment like the edge, we need to change the way we think about model serving. As we will describe in the following sections, the design of such a system is not trivial and there are several unsolved research challenges.

2.2 An Industrial Example

Modern manufacturing systems are composed of heterogeneous devices with varying levels of computing and storage capabilities. This can range from fixed-function micro-controllers with limited programmability to full-fledged servers. This diversity forces highly customized per-instance data system implementations based on available computing resources, network bandwidth, and assumptions about the future workloads. System designers must weigh the trade-offs between maintaining data near the edge where computing and storage resources are limited with the cost of transmitting data to more powerful computing systems. The introduction of new sensors, new robots, and new computers may invalidate previous assumptions that were used. These modifications can introduce communication and computation bottlenecks that require the redesign of the entire schema.

For instance, Audi has a business need to inspect welding robots used in the vehicle manufacturing process. The Audi plant in Neckarsulm, Germany contains 2,500 robots, some of which carry welding guns that perform a total of 5,000 welds per vehicle [8]. A machine learning model that considers images and sensor data from the assembly lines can be used to determine faults. This prediction task is latency-sensitive, and needs to consider priorities and assembly line deadlines. A systematic approach is needed that can optimize the data movement to ensure that tasks are completed within specified time thresholds. This includes determining where features should be computed, where predictions should occur, and how to adapt to network conditions.

2.3 Related Work

From academia to industry, people have been trying to make use of edge computing resources and move computing closer to users for a long time [37, 47, 56, 61]. There are already tens of edge computing frameworks targeting specific types of applications (video analytics, smart home, VR/AR gaming, autonomous vehicle, machine learning), endpoint devices (smartphones, IoT devices, cameras, drones) and edge nodes (highly-specialized hardware) [1, 7, 14, 15, 21, 24, 26, 27, 33, 34, 41, 50, 52, 58–60, 63, 65–67]. We focus on the narrow but nascent application of machine learning model serving, and believe that there are interesting optimization opportunities within this application. For example, Neurosurgeon [29] splits a DNN into an *edge part* and a *cloud part* at the granularity of neural network layers. Extending similar types of optimizations to models that integrate data over multiple data sources and those that are spatially partitioned over a network will be interesting extensions.

2.3.1 Combining Stream Processing and Machine Learning Inference. In a sense, EdgeServe will need to integrate two well-studied classes of systems: distributed stream processing systems and machine learning inference frameworks. To the best of our knowledge, such a system is a missing piece in the literature. There are existing cloud-based model serving systems, such as Clipper [10], TensorFlow Serving [46], and InferLine[9]; however, they are not designed for heterogeneous edge environments. While recent systems like TensorFlow Lite Pro [11] do support edge model deployment, they assume the user has manually programmed all of the necessary data movement.

On the other hand, there are a number of stream processing systems that are designed for data routing on the edge [54, 64]. NebulaStream [64] proposes a multi-layer topology to separate edge nodes *(fog layer)* from data collection points *(sensor layer)* and remote computational capability *(cloud layer)*. Routing nodes are network-aware and responsible for data transfer between all edge nodes. This architecture echoes early ideas from the HiFi sensor network architecture [17]. As the database community learned, optimizing data routing is an important part of distributed stream processing. For example, computing aggregate statistics over data from multiple different sensors requires smart communication strategies, such as routing trees in the TinyDB project [13, 38]. In a modern implementation, we can leverage more recent message-broker systems, such as RabbitMQ [48] and Kafka [4], and stream processing systems, such as Apache Spark Streaming [2], Apache Flink [3] and Apache Storm [5]. However, such systems are more tuned toward a cloud environment.

2.3.2 Routing and Management. A key aspect of EdgeServe will be a cost-based optimizer to make data movement and model placement decisions. Existing work covers various types of scheduling mechanisms for an edge network. Nebula [28] and NebulaStream [64] have a basic routing model where pre-specified nodes route data through the system. EdgeWise [18] incorporates a congestion-aware scheduler into a stream processing engine (SPE) and achieves better performance than traditional One Worker Per Operation Architecture (OWPOA) SPEs such as Apache Storm [5]. Frontier [45] uses a network-aware backpressure stream routing algorithm to determine transmission rates and dynamically route data downstream based on network path conditions. DPaxos [43] is a Paxos-based distributed protocol for data management across globally distributed edge nodes, which divides such nodes into regional zones to reduce latency within the same zone. Tetrium [23] proposes a heuristic to tackle the multi-resource (compute and network) allocation problem across heterogeneous geo-distributed clusters. None of them addressed the trade-off between system metrics and prediction accuracy, an important metric in model serving, as we will do in Section 3.2.

3 EDGESERVE: THE MISSING PIECE IN MACHINE LEARNING INFERENCE

The primary goal of EdgeServe is to facilitate machine learning inference on the edge where data sources may have to be *routed through the network before prediction.* Unlike existing model-serving systems, EdgeServe also orchestrates how data moves through the edge network. We envision a declarative system: the underlying data movement and placement policy should be automatically determined given a task specification and network capabilities.

3.1 Overview

We assume that each edge *node* is connected to others on a TCP/IP network (either directly or via a switched network). A subset of these nodes are physically connected to data sources (e.g. video cameras, sensors, and other data streams). Every node maintains a globally-synchronized catalog of data streams that are locally collected. EdgeServe is declarative in the sense that it decouples physical data collection with processing: decision making need not happen on the node collecting the data.

In EdgeServe, *models* are functions that are repeatedly applied to fixed windows of data. We assume white-box access to the models (all the parameters) and any feature transformations that need to be made before inference. Every model in EdgeServe has *locality constraints*, which describe where a model's prediction results have to be delivered. For example, one could require that both node1 and node2 need to have the output of the home activity recognition model above. In this case, once the inference is made on node1, the result must be streamed back to node2 (which is just another flow). However, not all requirements have to be this rigid, and we could require that either node1 or node2 has the required output. In this case, the extra flow is unnecessary. This class of requirements can be represented as a Boolean condition, such as "node 1", "any of node 1,4,5", or "all nodes 4,9,8".

Unlike existing model-serving systems that work asynchronously, EdgeServe works in a push-based model, where the arrival of each new data batch (defined by the user's inference task) triggers re-evaluation. These tasks subscribe to a message-broker service, which informs each node about new data. Each model can consume one or more sources of data and yields a new stream (a prediction). Since there is a global message-broker service, the output of models can be streamed to other models as well.

Models that consume multiple streams of data induce additional locality constraints, where data streams from multiple nodes may have to be aggregated in a central place. For example, an activity recognition model that requires video, audio, and network data must aggregate all of the data in a single place somewhere in the network. At a high level, EdgeServe combines a publication-subscription system to facilitate communication between multiple model-serving nodes on a network. To the best of our knowledge, such a system does not exist in part due to the challenges in routing, scheduling, and placement. The key system goal of EdgeServe is to provide a centralized control-plane to find placement, routing decisions, and model partitioning decisions that satisfy locality and hardware constraints.

3.2 Optimization Objectives

The main performance metric that we care about is *timeliness*, which is the time delay between data arrival and the prediction results arriving at the appropriate node in the edge network (independent of exactly where and how EdgeServe chose to execute that process).

Time-to-Decision (Timeliness). Unlike existing model serving systems where predictions are triggered by client requests, EdgeServe will continuously process predictions over streams of incoming data. Therefore, the concept of "latency" is a little more complicated in this setting. Accordingly, we define a new metric called timeliness, which is the gap between the time at which the data arrived and when a prediction was issued. The start time is defined as the time point at which all of the relevant data for a particular prediction is available somewhere on the network, and the end time is the time at which the prediction is issued and communicated to the appropriate edge node that can use the prediction.

The focus on model-serving makes the design of EdgeServe particularly interesting, because optimization decisions that improve the timeliness of predictions may affect their accuracy:

(1) *Prediction Accuracy (Accuracy).* We also care about the accuracy of the predictions that are made, or the gap between the prediction of a class or a continuous label and (hypothetical) ground-truth.
(2) *Robustness to Failure (Robustness).* Finally, it is also important to consider robustness to network and node failures. These failures can affect both the placement of computation, and the availability of source data. Robustness is measured in terms of the number and type of edge nodes that can be lost while still issuing a prediction.

All three of these metrics have both systems and machine learning implications. For example, there are systems solutions to improving timeliness through batching and locality, but there are also machine learning solutions where different model types have latency characteristics. Similarly, systems techniques like replication can help tolerate failures, but robust machine learning techniques can also allow for issuing predictions even if some of the features are lost.

3.3 Architecture

Next, we overview the architecture and implementation of EdgeServe. As depicted in Figure 2, there are 4 main components in our proposed system: (1) Data Input, (2) Optimizer, (3) EdgeServe Execution Engine, and (4) Cloud Synchronization.

(1) Data Input. EdgeServe will provide a domain-specific language (DSL) to describe model-serving tasks on the edge. The user will describe which nodes collect data, how those nodes are connected, the trained models to serve, and the endpoints at which model predictions should be delivered. EdgeServe will run as a persistent daemon on every node in the edge network and will discover network and

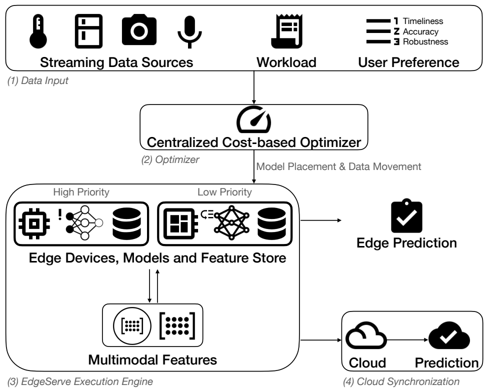

Figure 2: Four main components in EdgeServe

compute capabilities online. To support multi-tenancy, the user can also provide hints in terms of the desired timeliness, accuracy and robustness (defined above) of each model's predictions.

(2) Centralized Cost-based Optimizer. An optimizer parses the specifications from *(1) Data Input* and determines where best to place models and how to route data through the network. The optimizer needs to satisfy every model inference task's locality constraints while optimizing for a timely, accurate inference. This involves weighing communication costs, contention, as well as any specialized hardware, to accelerate prediction while maintaining accuracy. For example, if a node on the network has an attached GPU, it may be beneficial to stream image/video data to that node and results back rather than a local prediction. Accordingly, the optimizer will have to periodically re-optimize this global plan to account for shifting/bursty workloads and changing network conditions.

Architecturally, one node in the edge network is designated as a "leader" who hosts the centralized scheduler. This node also hosts a global catalog for the message-broker service giving every node on the network global visibility of all of the produced data streams. In our initial prototype, we do not handle leader failures.

This optimizer is highly related to similar proposals for Wide-Area Analytics (such as Tetrium) [23]. While such systems tackle communication constraints, data locality, and some amount of heterogeneity, they are incomplete in our context. In EdgeServe, not only is there consideration for data locality, there is also consideration for *result locality*. In edge-decision systems, prediction results may have to be delivered to a particular node on the network (e.g., the node that actuates a real-world system). The combination of data-locality and result-locality leads to a more challenging routing problem. Furthermore, our optimizer has to account for the location of machine learning accelerators which are not relevant for systems like [23].

(3) EdgeServe Execution Engine. All data in EdgeServe are interfaced through the message-broker system. Raw data streams are producers on the network, and model inference tasks are consumers on the network. The results of each model inference process are then new producers on the network (which other models can subscribe to). This architecture creates a naturally adaptive "push-based" task graph that is independent of where data are produced. Nodes simply need to know what streams to subscribe to and the leader informs the nodes where that resource is located. Nodes continuously serve model predictions over data that stream to them. The global routing table is synthesized based on the result of our centralized optimizer.

(4) Cloud Synchronization. Eventually, for model training, update, and archive, data will have to be moved off the edge and to the cloud. EdgeServe is able to synchronize data with a cloud service. The user can send featurized (or otherwise anonymized) data to the cloud.

3.3.1 Example Execution. Let's consider the home activity recognition example in the introduction. There are three *streams* of data: audio, video, and network traffic. Audio and video are collected on node1 (an Intel Video Processing Embedded System) and network traffic is collected on node2 (a programmable wireless access point). We have a *model* which is a neural network that requires all three data sources to predict ongoing activities in the home. To issue such predictions, the system could create a data flow (via publication and subscription) that repeatedly transfers raw data from node2 to node1, and host a comprehensive model on node1. Alternatively, it could also featurize the network traffic data locally on node2 and only transfer pre-processed features to node1. node1 applies a pooling method to issue a prediction based on features from multiple sources. This allows the user to combine the sources of data for a richer prediction, as well as leverage the specialized prediction hardware on both nodes.

4 RESEARCH AGENDA

We believe that EdgeServe is a key gap in current edge machine learning systems. Conceptually, it's a straightforward concept: combining a distributed message-broker system with model-serving nodes. However, the optimal design of such a system is an unsolved research question in its own right. We also believe that there are a number of future challenges, for us and the community, that would arise once an initial prototype is built.

4.1 Design Challenges

First, there are several technical challenges in the design of EdgeServe.

Step 1. Declarative Design. Existing model-serving systems are conceptually easy to use for users. By combining such systems with a message-broker interface, EdgeServe adds significant programming complexity. We believe that EdgeServe will only be practical as a declarative system where users specify high-level constraints which the optimizer turns into an execution plan. The optimizer would have to parse the specification, determine all placement and flow configurations that meet the specification, and find one that optimizes latency and data transfer objectives. The heterogeneity of modern edge networks is exploited as data distribution should be proportional to the heterogeneous capabilities of highly-specialized hardware. This requires accurate, dynamic cost-modeling of contention, latency and system performance on all of the nodes of an edge network and how they may affect accuracy. We know that this will be a formidable research challenge because in heterogeneous data center environments there has been already been significant work in task scheduling [12, 57, 62]. An optimizer for EdgeServe would have to address all of the problems in those works, as well as new ones that arise due to communication limitations.

Step 2. Fault Tolerance. Failures are an important characteristic of the edge, and fault-tolerance will have to be a primary design consideration for EdgeServe [51]. We envision a framework that ensures the availability of both data streams and model predictions in the presence of failures. The global routing table can be used to replicate both data and predictions to ensure that they are not lost. Similarly, redundant predictions can be generated by placing a model on multiple nodes. Therefore, in addition to optimizing for the user-specified constraints, we believe that the optimizer should additionally synthesize the appropriate replication strategy to meet availability constraints on data streams and predictions.

One interesting future research opportunity is to explore low-resolution replicas. Instead of exactly replicating a stream of data one could consider subsample or compress the stream. This would reduce the amount of data transfer but sacrifice accuracy in the event of a failure. Both the theory and practice of optimizing such a system are unknown and it would be an exciting future opportunity.

Step 3. Multi-Tenancy and Work Sharing. Furthermore, the edge generally does not have the same scale-out properties as the cloud. Thus, any edge computing framework will have to reason about how multiple users may share the same finite resources [35, 53]. In the machine learning context, multi-tenancy creates a number of new opportunities for work-sharing. For example, prior work has shown that combining redundant data streams can significantly improve the performance of Deep Learning workloads [36, 42], e.g., features created by one user can be reused by others. In addition to work-sharing, there are further challenges in meeting varying latency demands in a multi-user environment.

Step 4. Security and Debugging. As real-time decision systems get deployed in safety-critical applications such as industrial monitoring, transportation, and security, we will need to be able to retroactively audit such systems to explain their behavior. Containerized, black-box deployments of programs (as in existing systems) are opaque and hard to reason about. In contrast, we believe that EdgeServe would allow for detailed logging of what data is used for a decision and where it comes from. We envision a system that can track the provenance of the data and of the models deployed to be able to explain failures and other anomalous behavior. This logging system will periodically synchronize its data with a cloud service to enable model evaluation, retraining, and model updates.

4.2 Future Opportunities

Once built, we believe that EdgeServe will open up new research questions on the intersection of systems and machine learning. Accuracy and model design are particularly interesting optimization knobs in this context. Most AutoML frameworks today largely optimize for model accuracy without regard to the deployment environment. EdgeServe gives us a way to systematically consider timeliness, communication constraints and robustness in a single deployment framework. One could envision future AutoML frameworks searching for models with certain timeliness properties but also robustness and network communication properties as well. These frameworks would find the most accurate models and also have timely inference on the hardware in a particular edge network.

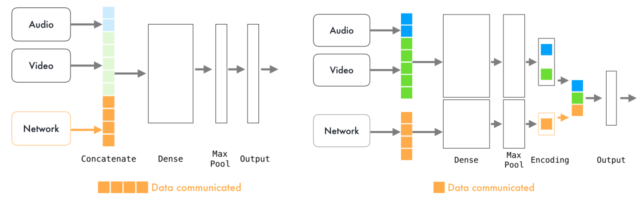

Figure 3: Some multimodal model architectures are more communication efficient than others, for example integrating data sources at later layers of a network.

Open Question 1. Communication-Efficient Models. There are a number of optimization opportunities in serving multi-modal predictions. At the most basic level, any time a model needs to integrate two different streams of data, communication between the respective sources is necessary. However, instead of communicating raw data streams, it may be far more efficient to communicate lower-dimensional features instead. Many machine learning models internally compress data into lower-dimensional representations and are highly robust to compressed inputs [16].

Figure 3 shows how the model structure plays an intimate role in how much data are communicated to serve the prediction. On the left, we have a model (neural network) that integrate the three data sources at the input layer. On the right, we have a model that integrates the data at a later layer. The model on the right is potentially more communication efficient than the one on the left. We believe that it is possible to integrate such constraints into the model search process–for example, by only considering neural network architectures with a certain communication cost. We can further optimize this process to construct a model in such a way that the features are spare using techniques like L1 regularization. Balancing these constraints with accuracy will be an important future research challenge.

Open Question 2. Naturally Fault-Tolerant Models. Another opportunity is to integrate fault-tolerance into the model architecture itself. Certain model architectures are naturally robust to losing one or more input sources. For example, consider the home activity recognition task before. Suppose that instead of a single architecture unified model, each data source was processed in its own model to issue a prediction using only the information within the model. These local models could be ensembled together (e.g, with a majority vote) to integrate the information from local predictions. On the other hand, a single unified model may not be able to tolerate the full loss of an input.

While the ensembling approach may lose correlations that occur across data sources, it can serve predictions even after losing input data sources. We believe that this indicates an efficiency-robustness trade-off in multimodal models, where some overall accuracy can be traded for robustness to the inputs. In general, we believe there is an interesting research direction to explore multimodal prediction architectures that ensure the result is at least as accurate as each constituent source.

ACKNOWLEDGMENTS

This work was supported by Intel and UChicago CERES.

REFERENCES

[1] Brian Amento, Robert J Hall, Kaustubh Joshi, and K Hal Purdy. 2017. Focusstack: Orchestrating edge clouds using focus of attention. *IEEE Internet Computing* 21, 1 (2017), 56–61.

[2] Apache. 2016. Apache Spark. https://spark.apache.org/ (visited on July 27, 2021).

[3] Apache. 2021. Apache Flink. Stateful Computations over Data Streams. https://flink.apache.org/ (visited on July 27, 2021).

[4] Apache. 2021. Apache Kafka. An open-source distributed event streaming platform. https://kafka.apache.org/ (visited on July 27, 2021).

[5] Apache. 2021. Apache Storm. An open source distributed realtime computation system. https://storm.apache.org/ (visited on July 27, 2021).

[6] Sirish Chandrasekaran and Michael J Franklin. 2003. PSoup: a system for streaming queries over streaming data. *The VLDB Journal* 12, 2 (2003), 140–156.

[7] Sharon Choy, Bernard Wong, Gwendal Simon, and Catherine Rosenberg. 2014. A hybrid edge-cloud architecture for reducing on-demand gaming latency. *Multimedia systems* 20, 5 (2014), 503–519.

[8] Intel Corporation. 2020. Audi's Automated Factory Moves Closer to Industry 4.0. https://www.intel.com/content/dam/www/public/us/en/documents/case-studies/audis-automated-factory-closer-to-industry-case-study.pdf (visited on July 27, 2021).

[9] Daniel Crankshaw, Gur-Eyal Sela, Xiangxi Mo, Corey Zumar, Ion Stoica, Joseph Gonzalez, and Alexey Tumanov. 2020. InferLine: latency-aware provisioning and scaling for prediction serving pipelines. In *SoCC '20: ACM Symposium on Cloud Computing, Virtual Event, USA, October 19-21, 2020*, Rodrigo Fonseca, Christina Delimitrou, and Beng Chin Ooi (Eds.). ACM, 477–491. https://doi.org/10.1145/3419111.3421285

[10] Daniel Crankshaw, Xin Wang, Giulio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. 2017. Clipper: A Low-Latency Online Prediction Serving System. In *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*, Aditya Akella and Jon Howell (Eds.). USENIX Association, 613–627. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/crankshaw

[11] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Shlomi Regev, et al. 2021. Tensorflow lite micro: Embedded machine learning on tinyml systems. In *Proceedings of Machine Learning and Systems 2021, MLSys 2021, San Jose, CA, USA, April, 2021*, Inderjit S. Dhillon, Dimitris S. Papailiopoulos, and Vivienne Sze (Eds.). mlsys.org.

[12] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. 2007. Dynamo: Amazon's highly available key-value store. *ACM SIGOPS operating systems review* 41, 6 (2007), 205–220.

[13] Alan Demers, Johannes Gehrke, Rajmohan Rajaraman, Niki Trigoni, and Yong Yao. 2003. The cougar project: a work-in-progress report. *ACM Sigmod Record* 32, 4 (2003), 53–59.

[14] Utsav Drolia, Katherine Guo, and Priya Narasimhan. 2017. Precog: prefetching for image recognition applications at the edge. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing, San Jose / Silicon Valley, SEC 2017, CA, USA, October 12-14, 2017*, Junshan Zhang, Mung Chiang, and Bruce M. Maggs (Eds.). ACM, 17:1–17:13. https://doi.org/10.1145/3132211.3134456

[15] Utsav Drolia, Katherine Guo, Jiaqi Tan, Rajeev Gandhi, and Priya Narasimhan. 2017. Cachier: Edge-Caching for Recognition Applications. In *37th IEEE International Conference on Distributed Computing Systems, ICDCS 2017, Atlanta, GA, USA, June 5-8, 2017*, Kisung Lee and Ling Liu (Eds.). IEEE Computer Society, 276–286. https://doi.org/10.1109/ICDCS.2017.94

[16] Adam Dziedzic, John Paparrizos, Sanjay Krishnan, Aaron Elmore, and Michael Franklin. 2019. Band-limited training and inference for convolutional neural networks. In *International Conference on Machine Learning*. PMLR, 1745–1754.

[17] Michael J. Franklin, Shawn R. Jeffery, Sailesh Krishnamurthy, Frederick Reiss, Shariq Rizvi, Eugene Wu, Owen Cooper, Anil Edakkunni, and Wei Hong. 2005. Design Considerations for High Fan-In Systems: The HiFi Approach. In *Second Biennial Conference on Innovative Data Systems Research, CIDR 2005, Asilomar, CA, USA, January 4-7, 2005, Online Proceedings*. www.cidrdb.org, 290–304. http://cidrdb.org/cidr2005/papers/P24.pdf

[18] Xinwei Fu, Talha Ghaffar, James C. Davis, and Dongyoon Lee. 2019. EdgeWise: A Better Stream Processing Engine for the Edge. In *2019 USENIX Annual Technical Conference, USENIX ATC 2019, Renton, WA, USA, July 10-12, 2019*, Dahlia Malkhi and Dan Tsafrir (Eds.). USENIX Association, 929–946. https://www.usenix.org/conference/atc19/presentation/fu

[19] Google. 2021. Edge TPU. https://cloud.google.com/edge-tpu (visited on July 27, 2021).

[20] Ramesh Govindan, Joseph Hellerstein, Wei Hong, Samuel Madden, Michael Franklin, and Scott Shenker. 2002. *The sensor network as a database*. Technical Report. Citeseer.

[21] Giulio Grassi, Kyle Jamieson, Paramvir Bahl, and Giovanni Pau. 2017. Parkmaster: An in-vehicle, edge-based video analytics service for detecting open parking spaces in urban environments. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. 1–14.

[22] Song Han, Huizi Mao, and William J. Dally. 2016. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

[23] Chien-Chun Hung, Ganesh Ananthanarayanan, Leana Golubchik, Minlan Yu, and Mingyang Zhang. 2018. Wide-area analytics with multiple resources. In *Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018*, Rui Oliveira, Pascal Felber, and Y. Charlie Hu (Eds.). ACM, 12:1–12:16. https://doi.org/10.1145/3190508.3190528

[24] Chien-Chun Hung, Ganesh Ananthanarayanan, Peter Bodik, Leana Golubchik, Minlan Yu, Paramvir Bahl, and Matthai Philipose. 2018. Videoedge: Processing camera streams using hierarchical clusters. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 115–131.

[25] Intel. 2021. Intel Movidius Vision Processing Units. https://www.intel.com/content/www/us/en/products/processors/movidius-vpu.html (visited on July 27, 2021).

[26] Minsung Jang, Karsten Schwan, Ketan Bhardwaj, Ada Gavrilovska, and Adhyas Avasthi. 2014. Personal clouds: Sharing and integrating networked resources to enhance end user experiences. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2220–2228.

[27] Si Young Jang, Yoonhyung Lee, Byoungheon Shin, and Dongman Lee. 2018. Application-aware IoT camera virtualization for video analytics edge computing. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 132–144.

[28] Albert Jonathan, Mathew Ryden, Kwangsung Oh, Abhishek Chandra, and Jon B. Weissman. 2017. Nebula: Distributed Edge Cloud for Data Intensive Computing. *IEEE Trans. Parallel Distributed Syst.* 28, 11 (2017), 3229–3242. https://doi.org/10.1109/TPDS.2017.2717883

[29] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor N. Mudge, Jason Mars, and Lingjia Tang. 2017. 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, ASPLOS 2017, Xi'an, China, April 8-12, 2017*, Yunji Chen, Olivier Temam, and John Carter (Eds.). ACM, 615–629. https://doi.org/10.1145/3037697.3037698

[30] Péter Kiss, Anna Reale, Charles Jose Ferrari, and Zoltán Istenes. 2018. Deployment of IoT applications on 5G edge. In *2018 IEEE International Conference on Future IoT Technologies (Future IoT)*. IEEE, 1–9.

[31] Sanjay Krishnan, Adam Dziedzic, and Aaron J. Elmore. 2019. DeepLens: Towards a Visual Data Management System. In *9th Biennial Conference on Innovative Data Systems Research, CIDR 2019, Asilomar, CA, USA, January 13-16, 2019, Online Proceedings*. www.cidrdb.org. http://cidrdb.org/cidr2019/papers/p40-krishnan-cidr19.pdf

[32] Philip Levis, Samuel Madden, Joseph Polastre, Robert Szewczyk, Kamin Whitehouse, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, et al. 2005. TinyOS: An operating system for sensor networks. In *Ambient intelligence*. Springer, 115–148.

[33] Yong Li and Wei Gao. 2018. MUVR: Supporting multi-user mobile virtual reality with resource constrained edge cloud. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 1–16.

[34] Peng Liu, Dale Willis, and Suman Banerjee. 2016. Paradrop: Enabling lightweight multi-tenancy at the network's extreme edge. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 1–13.

[35] Peng Liu, Dale Willis, and Suman Banerjee. 2016. Paradrop: Enabling lightweight multi-tenancy at the network's extreme edge. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 1–13.

[36] Rui Liu, Sanjan Krishnan, Aaron J Elmore, and Michael J Franklin. 2021. Understanding and Optimizing Packed Neural Network Training for Hyper-Parameter Tuning. In *Proceedings of the 5th International Workshop on Data Management for End-to-End Machine Learning, DEEM@SIGMOD 2021, June 20, 2021*. ACM.

[37] Yong Liu, Yang Guo, and Chao Liang. 2008. A survey on peer-to-peer video streaming systems. *Peer-to-peer Networking and Applications* 1, 1 (2008), 18–28.

[38] Samuel Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. 2002. TAG: A tiny aggregation service for ad-hoc sensor networks. *ACM SIGOPS Operating Systems Review* 36, SI (2002), 131–146.

[39] Samuel R Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. 2005. TinyDB: An acquisitional query processing system for sensor networks. *ACM Transactions on database systems (TODS)* 30, 1 (2005), 122–173.

[40] Sumit Maheshwari, Wuyang Zhang, Ivan Seskar, Yanyong Zhang, and Dipankar Raychaudhuri. 2019. EdgeDrive: Supporting advanced driver assistance systems using mobile edge clouds networks. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 1–6.

[41] Seyed Hossein Mortazavi, Mohammad Salehe, Carolina Simoes Gomes, Caleb Phillips, and Eyal de Lara. 2017. Cloudpath: A multi-tier cloud computing framework. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. 1–13.

[42] Deepak Narayanan, Keshav Santhanam, Amar Phanishayee, and Matei Zaharia. 2018. Accelerating deep learning workloads through efficient multi-model execution. In *NeurIPS Workshop on Systems for Machine Learning*. 20.

[43] Faisal Nawab, Divyakant Agrawal, and Amr El Abbadi. 2018. DPaxos: Managing Data Closer to Users for Low-Latency and Mobile Applications. In *Proceedings of the 2018 International Conference on Management of Data* (Houston, TX, USA) *(SIGMOD '18)*. Association for Computing Machinery, New York, NY, USA, 1221–1236. https://doi.org/10.1145/3183713.3196928

[44] NVIDIA. 2021. Nvidia Jetson. https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/ (visited on July 27, 2021).

[45] Dan O'Keeffe, Theodoros Salonidis, and Peter Pietzuch. 2018. Frontier: Resilient edge processing for the internet of things. *Proceedings of the VLDB Endowment* 11, 10 (2018), 1178–1191.

[46] Christopher Olston, Fangwei Li, Jeremiah Harmsen, Jordan Soyke, Kiril Gorovoy, Li Lao, Noah Fiedel, Sukriti Ramesh, and Vinu Rajashekhar. 2017. TensorFlow-Serving: Flexible, High-Performance ML Serving. In *Workshop on ML Systems at NIPS 2017*.

[47] Trang Quang and Yang Peng. 2020. Device-driven On-demand Deployment of Serverless Computing Functions. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 1–6.

[48] RabbitMQ. 2021. An open-source message-broker software. https://www.rabbitmq.com/ (visited on July 27, 2021).

[49] Research and Markets. 2021. Public Edge Computing: the Opportunity for Third-Party CDN Providers. https://www.researchandmarkets.com/reports/5239264/public-edge-computing-the-opportunity-for-third (visited on July 27, 2021).

[50] Hooman Peiro Sajjad, Ken Danniswara, Ahmad Al-Shishtawy, and Vladimir Vlassov. 2016. Spanedge: Towards unifying stream processing over central and near-the-edge data centers. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 168–178.

[51] Mahadev Satyanarayanan. 2017. The emergence of edge computing. *Computer* 50, 1 (2017), 30–39.

[52] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. 2009. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing* 8, 4 (2009), 14–23.

[53] Mennan Selimi, Adisorn Lertsinsrubtavee, Arjuna Sathiaseelan, Llorenc Cerda-Alabern, and Leandro Navarro. 2019. PiCasso: Enabling information-centric multi-tenancy at the edge of community mesh networks. *Computer Networks* 164 (2019), 106897.

[54] Zhitao Shen, Vikram Kumaran, Michael J. Franklin, Sailesh Krishnamurthy, Amit Bhat, Madhu Kumar, Robert Lerche, and Kim Macpherson. 2015. CSA: Streaming Engine for Internet of Things. *IEEE Data Eng. Bull.* 38, 4 (2015), 39–50. http://sites.computer.org/debull/A15dec/p39.pdf

[55] Statista. 2021. Smart home installed devices U.S. 2018-2023. https://www.statista.com/statistics/1075749/united-states-installed-base-of-smart-home-systems/ (visited on July 27, 2021).

[56] Animesh Trivedi, Lin Wang, Henri E. Bal, and Alexandru Iosup. 2020. Sharing and Caring of Data at the Edge. In *3rd USENIX Workshop on Hot Topics in Edge Computing, HotEdge 2020, June 25-26, 2020*, Irfan Ahmad and Ming Zhao (Eds.). USENIX Association. https://www.usenix.org/conference/hotedge20/presentation/trivedi

[57] Alexey Tumanov, James Cipar, Gregory R Ganger, and Michael A Kozuch. 2012. alsched: Algebraic scheduling of mixed workloads in heterogeneous clouds. In *Proceedings of the third ACM Symposium on Cloud Computing*. 1–7.

[58] Junjue Wang, Ziqiang Feng, Zhuo Chen, Shilpa George, Mihir Bala, Padmanabhan Pillai, Shao-Wen Yang, and Mahadev Satyanarayanan. 2018. Bandwidth-efficient live video analytics for drones via edge computing. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 159–173.

[59] Zhi-Wei Xu. 2014. Cloud-sea computing systems: Towards thousand-fold improvement in performance per watt for the coming zettabyte era. *Journal of Computer Science and Technology* 29, 2 (2014), 177–181.

[60] Shanhe Yi, Zijiang Hao, Qingyang Zhang, Quan Zhang, Weisong Shi, and Qun Li. 2017. Lavea: Latency-aware video analytics on edge computing platform. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. 1–13.

[61] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P Jue. 2019. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture* 98 (2019), 289–330.

[62] Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy H. Katz, and Ion Stoica. 2008. Improving MapReduce Performance in Heterogeneous Environments. In *8th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2008, December 8-10, 2008, San Diego, California, USA, Proceedings*, Richard Draves and Robbert van Renesse (Eds.). USENIX Association, 29–42. http://www.usenix.org/events/osdi08/tech/full_papers/zaharia/zaharia.pdf

[63] Xiao Zeng, Biyi Fang, Haichen Shen, and Mi Zhang. 2020. Distream: scaling live video analytics with workload-adaptive distributed edge intelligence. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. 409–421.

[64] Steffen Zeuch, Ankit Chaudhary, Bonaventura Del Monte, Haralampos Gavriilidis, Dimitrios Giouroukis, Philipp M. Grulich, Sebastian Breß, Jonas Traub, and Volker Markl. 2020. The NebulaStream Platform for Data and Application Management in the Internet of Things. In *10th Conference on Innovative Data Systems Research, CIDR 2020, Amsterdam, The Netherlands, January 12-15, 2020, Online Proceedings*. www.cidrdb.org. http://cidrdb.org/cidr2020/papers/p7-zeuch-cidr20.pdf

[65] Quan Zhang, Xiaohong Zhang, Qingyang Zhang, Weisong Shi, and Hong Zhong. 2016. Firework: Big data sharing and processing in collaborative edge environment. In *2016 Fourth IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*. IEEE, 20–25.

[66] Wuyang Zhang, Jiachen Chen, Yanyong Zhang, and Dipankar Raychaudhuri. 2017. Towards efficient edge cloud augmentation for virtual reality mmogs. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*. 1–14.

[67] Xu Zhang, Hao Chen, Yangchao Zhao, Zhan Ma, Yiling Xu, Haojun Huang, Hao Yin, and Dapeng Oliver Wu. 2019. Improving cloud gaming experience through mobile edge computing. *IEEE Wireless Communications* 26, 4 (2019), 178–183.