Salient Store: Enabling Smart Storage for Continuous Learning Edge Servers

Cyan Subhra Mishra, Deeksha Chaudhary, Jack Sampson, Mahmut Taylan Knademir, Chita Das The Pennsylvania State University (cyan, dmc6955, jms1257, mtk2, cxd12)@psu.edu

Abstract

As continuous learning based video analytics continue to evolve, the role of efficient edge servers in efficiently managing vast and dynamic datasets is becoming increasingly crucial. Unlike their compute architecture, storage and archival system for these edge servers has often been under-emphasized. This is unfortunate as they contribute significantly to the data management and data movement, especially in a emerging complute landscape where date storage and data protection has become one of the key concerns. To mitigate this, we propose Salient Store that specifically focuses on the integration of Computational Storage Devices (CSDs) into edge servers to enhance data processing and management, particularly in continuous learning scenarios, prevalent in fields such as autonomous driving and urban mobility. Our research, gos beyond the compute domain, and identifies the gaps in current storage system designs. We proposes a framework that aligns more closely with the growing data demands. We present a detailed analysis of data movement challenges within the archival workflows and demonstrate how the strategic integration of CSDs can significantly optimize data compression, encryption, as well as other data management tasks, to improve overall system performance. By leveraging the parallel processing capabilities of FPGAs and the high internal bandwidth of SSDs, Salient Store reduces the communication latency and data volume by $\approx 6.2 \times$ and $\approx 6.1 \times$, respectively. This paper provides a comprehensive overview of the potential of CSDs to revolutionize storage, making them not just data repositories but active participants in the computational process.

1 Introduction

Video analytics, powered by deep neural networks (DNNs) has become the key component of multiple applications including but not limited to autonomous driving (Brown et al., 2023; Fang et al., 2023; Huang et al., 2023), urban mobility (Corporation; Custom On-Device ML Models with Learn2Compress), surveillance and monitoring (Bozcan & Kayacan, 2020; Dutta & Ekenna, 2019; Pichierri et al., 2023), video streaming and conferencing (Dasari et al., 2022b; Cheng et al., 2024; Sivaraman et al., 2024), telemedicine (Wan et al., 2020), and tourism (Zhu et al., 2024; Pierdicca et al., 2021; Godovykh et al., 2022). While some of these applications rely on collecting the video data and processing them offline, many need real-time analytics for the seamless integration, operation and effectiveness of the task at hand (Bramberger et al., 2004; Apostolo et al., 2022; Grulich & Nawab, 2018). Moreover, depending on the deployment, scenario and requirements, some of these applications also demand learning to keep up with the data drift (Bhardwaj et al., 2022; Mishra et al., 2024; Kim et al., 2024; Rebuffi et al., 2017). However, regulations, resource limitations and privacy concerns often mandate these applications (both learning and inference) to be performed at the edge (Bhardwaj et al., 2022; Mishra et al., 2024). For example, many of the European cities restrict the traffic video data to be streamed to the cloud (www.dlapiperdataprotection.com; Achieving Compliant Data Residency and Security with Azure; Bhardwaj et al., 2022), which enforces performing video

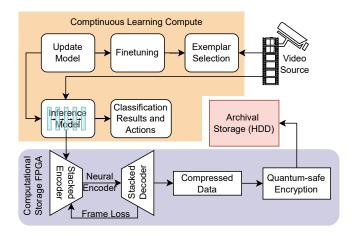


Figure 1: Data flow pipeline of continuous learning edge servers with storage and data archival pipeline. The Shown storage pipeline is the preliminary focus of $Salient\ Store$.

analytics and learning tasks related to urban mobility at the edge. This has lead to a significant development in the direction of enabling video analytics and learning with edge servers.

Although efficient algorithms, compute orchestration and hardware have addressed the analytics part, the scaling of such a system becomes a problem primarily due to high energy consumption. Recent works try to solve this problem by augmenting these continuous learning edge servers with application-specific hardware targeted for intermittent computing which could run using solar power. However, all these works focus on the analytics part while overlooking one critical aspect: what happens to all those video data after the analytics?

<u>Data Archival:</u> The answer is straightforward, especially for mission-critical public records like urban mobility and surveillance data: these need to be archived in a local storage to avoid undermining the benefits of edge computation, i.e. minimizing communication and preserving privacy. However, managing such data requires substantial storage infrastructure. For instance, storing a full day's worth of 1080p video at 60fps requires ($1920 \times 1080 \times 3$ pixels $\times 4$ bytes per pixel $\times 60$ frames per second $\times 3600 \times 24$)117.32 TiB of raw video data, which compresses to approximately 60 GiB to 400 GiB of encoded data per day. Including redundancy, this requires an additional 33% to 100% more storage capacity. Furthermore, given the plug-and-play nature of storage media like HDDs and SSDs, securing this data becomes even more critical.

The typical *archival process* involves three key phases: compression, encryption, and redundancy. Fig. 1 illustrates the data flow in an edge server, where video data are first encoded (using H264 or similar codecs), then encrypted (with RSA or similar standards), and finally stored across a distributed set of disks to ensure redundancy (e.g., RAID 5). These processes create a complex data-flow pipeline, differentiating two streams of video data: one for real-time inference and training, and another for archival. This dual-stream processing consumes considerable system resources such as CPU, memory, and energy, further complicating the management for intermittently powered systems like Uṣás (Mishra et al., 2024) where data integrity and security must be maintained during power disruptions ¹. Table 1 provides an estimation of resource utilization on commercial systems underscoring the fact that compressing, encrypting and reliably storing the data, especially for (intermittent) edge servers is a bigger challenge in contrast to classical cloud storage servers.

<u>The Challenges:</u> In edge computing architectures, the **lack of data reuse** between the analytics and video data archival poses significant challenges. Classically, video data is streamed simultaneously to compute and storage systems for various processing tasks, such as inference, exemplar selection, and storage, thereby increasing I/O bandwidth and system processing demands. This processing complexity necessitates substantial compute and memory resources, escalating power consumption

¹Management and retrieval of data typically utilize a vector database like file-system, although this is beyond the scope of this discussion.

Data	Task	Algorithm	% CPU Utilization	% DRAM Utilization	
			16 Core Xeon	Peak	Average
All	Encryptions	RSA512	2.18	14.56	5.85
All	Decryptions	RSA512	3.45	17.2	6.12
3D PC	Compression	OctTree	26.78	78.2	32.54
	Inflation	OctTree	29.24	81.56	36.18
Video	Compression	ZStd	24.7	62.54	24.5
	Inflation	ZStd	22.6	79.18	29.43
	Compression	H264	12.85	52.46	21.4
	Inflation	H264	14.2	69.46	26.18
All	(un)RAID	Unraid	11.25	29.4	19.24

Table 1: Resource utilization while running different algorithms under classical data archival pipeline for multiple data modalities in an AWS h1.4xlarge storage-optimized instance.

(e.g., systems with CPUs having a thermal design power of 145W and 64GB of DRAM as noted in Table 1) and requiring larger form factors, which exceed the capabilities of intermittent systems and challenge sustainability goals. Therefore, there is an urgent need to *minimize compute and power requirements* in archival processes. Moreover, the divergence of analytics and archival pipelines from the outset suggests that optimizing both hardware, software and data-flow used in inference, learning and archival could significantly enhance throughput and energy efficiency. This can be achieved by using modern *neural compression algorithms* instead of the classical encoding algorithm. However, the neural compression algorithm needs to be *compute efficient* (reusing maximum analytics pipeline), have *high compression ratio* (need to compete with H264) and *feature rich* (could be decompressed and retrieved with a reasonable loss). Furthermore, it needs to take advantage of the *data similarity between frames* to further minimize the storage footprint, and thereby reducing the form factor and need of frequent disk swapping/ maintenance.

The second challenge comes from **privacy and security of sensitive data**. Considering the cheap commodity use storage devices are often plug and play, they are often vulnerable for data leak, especially if they are deployed in public, like urban mobility setting. Unlike secure data centers, these federated, distributed and public deployment could be susceptible to direct physical attacks for data breach. Although modern encryption algorithms like RSA are secure, there is still a threat of *store now decrypt later*² kind of attack (National Cybersecurity Center of Excellence (NCCoE), 2023). To mitigate this, *quantum safe encryption algorithms needs to be used without hindering the throughput*. Furthermore, the design *needs to be programmable* to ensure encryption keys to be changed regularly for additional security.

Solution Space and Our Work: This paper proposes Salient Store, a novel storage solution designed for continuous learning edge servers by incorporating a hardware-software co-design framework that allows for efficient data archival and storage. Salient Store utilizes the state-of-the-art neural compression which partially uses the inference/exemplar selection pipeline along with layered neural codecs to compress the video data. It also uses the motion vectors as a latent space to effectively use the inter-frame similarity, thereby further increasing the compression ratio. Salient Store also provides a hardware accelerated lattice-based quantum safe encryption mechanism. To tightly integrate these solutions to the storage space, while providing data security, Salient Store uses computational storage devices (CSDs) (AMD, b) which reduce the energy consumption while keeping the compute pipeline unaltered. Our main contributions include:

We propose design of a hybrid storage pipeline equipped with computational storage drives (CSDs)
where the different drives could communicate with each other in a peer-to-peer fashion. These
CSDs synergistically orchestrate the archival related tasks between the storage controller CPU and
the computational storage FPGAs. The hybrid storage system is capable of taking the computed

²Given a powerful enough computer like quantum computers, RSA encrypted data can be decrypted, and therefore National Institute of Standards and Technology (NIST) called for proposals to develop post quantum cryptography algorithms (National Institute of Standards and Technology (NIST), 2024), and defined a standard for the same. One of the successful submissions – which was later defined as a standard – uses lattice-based encryption algorithm (Micciancio & Regev, 2009) and will be the focus of our work.

frame features and the motion vectors from the compute hardware to perform a novel layered neural compression.

- We discuss the storage data-flow, the compute orchestration and mapping in the proposed system.
 This includes a hardware software co-design for compute-intensive applications along with mapping different functions to different hardware in the data pipeline. Furthermore, we add failure management support for the intermittent edge servers.
- We go beyond the compute and look into future-proofing the storage server by equipping it with quantum safe lattice-based encryption technique. We maximize the hardware utilization by reusing compute kernels from the compression pipeline. We detail the design of the hardware accelerated encryption and maximize the resource reuse between the exemplar selection and encryption.
- Finally we perform an in-depth exploration of this integration, supported by real-world data from domains such as autonomous driving and urban mobility, to illustrate its effectiveness in continuous learning scenarios. The proposed design provides $\approx 2.2 \times$ latency and $\approx 5.6 \times$ data movement benefits compared to the state-of-the-art, on a single storage and $\approx 4.8 \times$ latency benefit in a multi-node system.

2 Background and Motivation

2.1 Storage for Continuous Learning Edge Servers

Recent developments in continuous learning for video analytics (Bhardwaj et al., 2022; Mishra et al., 2024; Kim et al., 2024) has significantly boosted the capabilities and accuracy of learning systems. The major focus of these works have been building compute platforms with efficient scheduling (Bhardwaj et al., 2022; Mishra et al., 2024), and reconfigurable hardware design (Kim et al., 2024; Mishra et al., 2024). This solves majority of the bottlenecks in a performance-driven classical cloud server platform. However, video analytics for many applications (Corporation; Custom On-Device ML Models with Learn2Compress; Wright"; "premioinc") are moving towards the edge. Therefore, managing and storing the hefty volume of video data brings more challenge due to the energy, compute and form-factor limitations. Modern and upcoming applications like urban mobility and autonomous driving are predicted to be generating hundreds of exabytes of data (Urban Traffic Dataset; Corporation; Wright"; "premioinc") per year while increasingly being deployed at the edge calling for a robust, secure, and efficient infrastructure for storing data at the edge while needing occasional human intervention for maintenance.

Using State-of-the-Art Video Data Storage? Maybe Not: While storing video data as files works for small systems, in very large-scale systems they are typically stored using vector databases (Shen et al., 2005; Pan et al., 2024). Vector databases typically extract features from the video data to form index and those indices are then sorted using various metrics like neighborhood, maximum similarity, etc. (Fonseca & Jorge, 2003; Cao et al., 2013; Tian et al., 2023; Douze et al., 2024) for faster retrieval. The vector index are used to point to the meta-data of the video file and then the actual video data is retrieved from the storage (Shen et al., 2005). This approach helps context-based search like looking for particular objects, events, or attributes (Douze et al., 2024). However, it is obvious that this approach, albeit good for streaming and retrieval, are not entirely space-efficient, and at times can increase the data volume by many folds (Douze et al., 2024). In an edge server where we are only worried about storage and not retrieval³, the vector database approach is not effective. Rather, storing the data in a compressed and encrypted format (with redundancies) is more efficient and therefore is the focus of our work.

Why Not the Usual Process? Now that we know Classical approach of video data storage involves encoding and encrypting the video data before storing them in a redundant storage array (Huang & Xu, 2014; Fan et al., 2022; Korkiakangas, 2014; Yue et al., 2016) which consumes significant resources (refer Table 1). Even though there have been significant research in accelerating both compression (Collet & Kucherawy, 2018; Chen et al., 2021) and encryption (Milanov, 2009; Rawat et al., 2019; Yang et al., 2015), operating on large-scale video data often demands more resource than what edge servers could afford (Mishra et al., 2024). Co-locating this compute along with the inference and training would definitely hinder the critical path.

³We assume the data to be eventually available in the data repository, where they can be properly stored for efficient lookup. This can be done by periodically transporting the data by swapping out storage bays. Our goal is to maximize storage at edge so that the frequency of maintenance decreases.

The main reason these algorithms consume significant resources is because of the amount of data they handle. Every single 1920×1080 raw frame prior to encoding carries $\approx 23 \text{MiB}$ of data which, @60fps, will require processing $\approx 1.4 \text{GiB}$ of data per second per imaging source. This pipeline assumes the use of classical data encoding algorithms like $H264^4$ which requires all the frames to encode a video stream, albeit it only saves the essential information. Therefore, it does not use any of the computations that are used in the inference and learning pipeline. Then, the question is: is there a way we can reuse the computations used for the inference to help us in encoding the data? The answer is *neural codecs* (Ma et al., 2019; Chen et al., 2017).

Neural Codecs – DNNs for Compression: Neural codecs represent a paradigm shift in video compression technology, leveraging the capabilities of deep learning to optimize both encoding and decoding processes. Unlike traditional codecs that rely on predefined algorithms to compress video data, neural codecs utilize an end-to-end trainable system based on neural networks. These networks are trained on extensive video datasets, allowing them to dynamically adapt compression strategies based on the content's complexity and prevailing network conditions. The architectural backbone of neural codecs typically comprises an autoencoder, where the encoder compresses the video into a compact, lower-dimensional representation, and the decoder reconstructs it back into video format. These blocks can be stacked over each other to form layered codecs (like SHVC and SVC). This process benefits significantly from residual learning techniques, where each successive layer in the network aims to correct errors from the previous layers, thereby enhancing the reconstructed video quality incrementally with each additional decoding layer.

Moreover, neural codecs excel in adaptability, offering robust performance across variable bandwidth and computational conditions, making them particularly suited for real-time streaming environments. These codecs can operate on generic computational hardware, such as GPUs and FPGAs, without the need for specialized video processing units, thus broadening their applicability across different device platforms. This adaptability also extends to content delivery dynamics, where neural codecs can adjust the streaming quality in real-time, responding adeptly to fluctuations in network throughput and variations in device capabilities. Such capabilities not only enhance user experience by minimizing buffering and maximizing video quality but also optimize bandwidth usage, presenting a cost-effective solution for content providers. These technical advancements position neural codecs as potential game-changers in the video streaming industry (NVIDIA Corporation, 2024), promising significant improvements in efficiency, scalability, and flexibility in various streaming scenarios.

One major issue with neural codecs are their lack of utilization of inter-frame similarity. Classically, neural codecs compress each frame by treating them like images. However, video data often comes with a large amount of inter-frame similarity (Ying et al., 2022b; Zhang et al., 2017; Zhao et al., 2020, 2021; Ying et al., 2022a). This similarity could be exploited, like in classical encoding algorithms, to further increase the compression ratio while improving the feature quality. Furthermore, neural codecs offer us the flexibility to approximate the computation (using quantization) to further enhance efficiency. Since they also use the computation blocks of the standard neural network, they can be jointly trained along with the classifier network to perform as the feature extraction and encoding backbone, thereby maximizing the utilization of the inference pipeline, and reducing the computation needed for compression. This **maximizes data and resource reuse**, and the pipeline for inference and compression do not diverge from the get going.

However, even with data reuse, performing compression using neural codecs would require extra compute resources, energy and latency, hindering the critical path, i.e., inference. However, *this issue could be alleviated by not using the compute resource in the critical path and preferably moving the compute to the storage where the data will eventually be stored.* This is where the modern and upcoming computational storage devices (CSDs) come to rescue: bringing computation closer to data while providing with maximum energy efficiency and programmability (Newsroom; AMD, b). However, storage systems are not typically built to cater towards the ML applications, and now that compression becomes a ML application with the use of stacked neural codecs, building the right storage stack along with computational storage devices becomes an important problem.

Evolution of Computational Storage: The advent of CSDs represents a paradigm shift, bringing computation closer to storage. These devices, by integrating CPUs or FPGAs into the storage

⁴We do *not* consider H265 here as currently in commercial systems H264 is the standard and typically enjoys hardware support. Moreover, H265 is an extension of H264 with additional features like coding tree units and intra-prediction directions which demand significantly more computation.

medium, facilitate computation at the storage level. Initial applications of CSDs were confined to tasks like encryption/decryption, RAID, and compression. However, there has been a significant push towards enabling more complex workloads, including query processing on CSDs. Efforts to adapt CSDs for machine learning applications, albeit limited in scope to classical learning and data management, mark a crucial step forward. Yet, the expansion of these technologies to encompass large-scale storage stacks and the integration of CSDs into conventional storage systems, particularly for ML applications, remains a significant challenge and an open area of research.

Bridging the Gap in Storage System Design: There is a discernible gap in the design and conceptualization of storage drives, systems and servers, especially in the context of ML applications. Historically, researchers have investigated these components individually, often focusing on high-performance computing, scientific computing, and database applications. However, the specific demands of ML applications on storage systems have largely been overlooked. storage stack architects often abstract the computational processes, neglecting considerations such as data movement cost, compute cost, and power requirements. Conversely, architects of storage drives, including CSDs, tend to overlook the broader application requirements, such as data prioritization, potential offloading of computational tasks to storage, and application-specific data compression and encryption strategies. Our research aims to bridge this gap, focusing specifically on the exigencies of continuous learning applications.

2.2 The Problem: Understanding the Data Flow

Challenges in Data Movement: In both consumer applications and high-performance computing (HPC) programs, the process of data collection followed by analytics is a critical operation (Cao et al., 2020; Mailthody et al., 2019; Chapman et al., 2019; Li et al., 2023). The efficiency of data movement significantly influences the performance and energy consumption of large-scale systems (Park et al.). As earlier discussions highlighted, applications on edge servers need to store generated data on storage stacks. These data are can then be moved to a central facility where they can be further treated for efficient retrieval ans on demand streaming.

Urban Mobility - A Case Study in Continuous Learning: To thoroughly understand this data flow, let us examine "urban mobility", a prevalent continuous learning task (Bhardwaj et al., 2022). This task dynamically adapts to changing traffic patterns and environmental conditions. In urban mobility, high-volume data streams from multiple sources, such as high-resolution traffic cameras, are first compressed then analyzed for exemplar selection. This process involves "representation learning" (Rebuffi et al., 2017), where data is transformed into "feature vectors" using deep neural networks, followed by unsupervised learning techniques like k-means clustering. The goal is to identify unique or new classes of data for training while archiving known classes. In exemplar selection, the entire dataset is analyzed to detect classes with unique features, i.e., the images that are much different from the training data distribution or new classes that were not included in the training data. This is typically achieved through representation learning (Rebuffi et al., 2017) where the data is first converted into a feature vectors using the convolution layers of a large DNN model (or multiples of them), and then performing an unsupervised learning based classification, e.g., k-means++ (Arthur & Vassilvitskii, 2007; Bahmani et al., 2012), to cluster the data. It is noteworthy that, the process of neural compression as well as inference/representation learning use the feature extraction method. In this work, our goal is to use this to our advantage and maximize the compute and data reuse.

The critical insight to be gleaned here is reusing data and compute pipeline prior to its transfer to storage could markedly diminish the costs associated with data movement. This approach would notably minimize the consumption of bandwidth and latency and curtail the compute requirements, thereby ensuring that the compute server's resources are judiciously employed only for indispensable computations. This strategic compute-reuse could help in optimizing the efficiency and efficacy of computational operations, especially in large scale data intensive and data driven applications.

2.3 Why Not More Compute at Storage Stacks?

Integrating additional compute capabilities within storage stacks to offload certain computational tasks may seem an intuitive solution to the problem at hand. Theoretically, storage stacks could handle operations like unRAID and decryption, and extend their functionality to data inflation and

feature extraction. This would ostensibly allow for the selective transfer of only pertinent exemplar data to compute servers, thereby optimizing data movement costs. However, this approach presents several challenges. Current storage stacks are outfitted with robust CPUs and memory systems, which are heavily tasked with state-of-the-art storage management algorithms, as evidenced by the resource utilization outlined in TABLE 1. These algorithms already consume substantial compute and memory resources, often to the extent of fully occupying the storage controller system, with a portion of resources being allocated for essential system stack operations. Furthermore, incorporating more advanced compute hardware, such as FPGAs, GPGPUs and other accelerators, would lead to underutilized I/O slots and memory, consequently escalating the cost of storage systems. Additionally, storage solutions that are optimized for I/O throughput tend to lose their specialized efficiency when repurposed for general compute tasks (Haynes et al., 2021). While the previous research has suggested the idea of integrating analytics into storage systems, these discussions usually revolve around system architecture, scheduling, and data management, without delving into the requisite compute capabilities (Haynes et al., 2021; Daum et al., 2021; Tsai et al., 2020; Xu et al., 2019).

2.4 More Compute in Storage? Why Not?

Exploring the realm of computational storage drives presents a tantalizing avenue for on-disk computing capabilities. Storage controllers, typically constrained by I/O bandwidth, are now being complemented by the vast internal bandwidth of solid-state drives (SSDs), making them prime candidates for near-data processing. Recent advances in both commercial (Newsroom; AMD, b; Mesnier, 2022; ScaleFlux, a,b; Eideticom & Laboratory; Laboratory) and academic sectors (Torabzadehkashi et al., 2019b; Barbalace & Do, 2021; Lukken & Trivedi, 2021; Torabzadehkashi et al., 2019a; Salamat et al., 2021, 2022; Do et al., 2013) advocate the use of computational storage drives (CSDs) across databases, high-performance computing (HPC), and analytics. AMD and Xilinx have introduced specialized tools and libraries designed to harness CSDs (AMD, a; AMD & Xilinx), enabling peer-to-peer PCIe transactions that bypass the CPU (AMD & Xilinx). The emergence of Compute Express Link (CXL) technology (Sharma, 2022a,b; Jung, 2022) further amplifies the potential of disaggregated storage and memory systems.

Decoupling compute tasks needed for storing the data from the host CPU and embedding them directly within storage devices, particularly through CSDs, has demonstrated significant performance and energy benefits. Tasks traditionally performed at the storage controller level are now being offloaded to CSDs, often accelerated using FPGA primitives (Kim et al., 2021; AMD & Xilinx; AMD, a). Moreover, CSDs hold the potential to undertake critical machine learning tasks like feature extraction and clustering, streamlining tasks like neural compression. The unique combination of FPGAs' parallel processing prowess, the substantial internal bandwidth of SSDs, and their block-accessible nature position CSDs as "ideal components" for evolving smart storage solutions tailored for machine learning.

To cater towards this, in this work, we propose Salient Store – a mini computational storage server (we call it "edge storage server") stack for managing the data archival in edge servers. Salient Store provides adaptive data compression using neural codecs and further enhances the data security by providing an accelerated quantum safe data encryption policy, protecting these vulnerable edge storage servers against the store now decrypt later (National Cybersecurity Center of Excellence (NCCoE), 2023) attacks.

3 Data Compression using Neural Codec

In this section we go over the overall design and design choices for the neural codec design of <code>Salient Store</code> . We discuss the edge storage architecture followed by the choice of neural codec and their design.

3.1 Salient Store Storage Architecture:

Salient Store edge storage architecture is crafted to optimize data-flow and computational efficiency in continuous learning edge servers. Salient Store employs a "hybrid model", integrating Computational Storage Drives (CSDs) with Field-Programmable Gate Arrays (FPGAs) (AMD, b) and classical storage drives. This design, depicted in Fig. 2, gives the programmable computational capa-

bilities of CSDs along with the cost-effectiveness and durability of the classical HDDs. It leverages the PCIe interface for efficient "peer-to-peer communication", substantially reducing communication latency and energy requirements. This setup alleviates the host CPU from handling frequent data movement interruptions. A storage platform entirely designed with CSDs is not pragmatic at the current time because of their exorbitant cost and power consumption (AMD, b; Cao et al., 2020). However, a combination of CSDs with classical storage medium provides the most optimal solution and hence motivates our design.

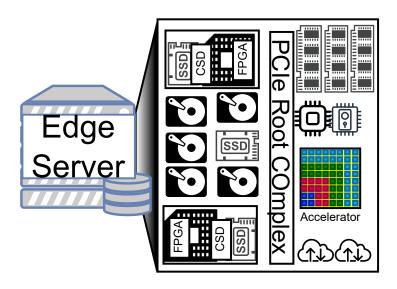


Figure 2: High-level design of the *Salient Store* edge server - it consists of the accelerated video analytics compute along with computational storage and classical storage drives.

Data-flow Reorganization in Salient Store: At the core of **Salient Store** 's design is the "data-aware" reorganization of compute processes. Unlike conventional storage servers, **Salient Store** discerns between the "data path" and the "resource path" for system I/O calls, translating these requests into CSD-specific functions. These functions leverage FPGA kernels for efficient data processing.

As discussed earlier, Salient Store edge storage implements a video archival by using neural compression followed by a quantum safe encryption (refer Fig. 1). To maximize the compute reuse between the compute pipeline and the archival pipeline, Salient Store uses apart of the neural network of the inference engine to extract features, and then further performs the encoding using the FPGA in the CSD.

Neural codecs present a new paradigm video compression technology, leveraging deep learning to surpass traditional codec efficiency (Ma et al., 2019; Chen et al., 2017). Traditional neural codecs, while innovative, typically encode and decode video streams in a monolithic fashion, which often results in suboptimal utilization of computational resources and inflexibility (Ma et al., 2019). This inherent inefficiency stems from their design which does not allow incremental improvements in video quality and often leads to either over-utilization or under-utilization of bandwidth. To address these shortcomings, the advent of layered neural codecs marks a significant advancement (Dasari et al., 2022a). Layered neural codecs, by design, encode video into multiple, distinct layers of data, each enhancing the video quality incrementally. This allows for dynamic adaptation to network fluctuations and client-side computational capabilities, thereby optimizing the streaming experience. The capability to adjust the quality dynamically, by decoding additional layers as resources permit, ensures efficient bandwidth usage and enhances the overall Quality of Experience (QoE), making layered neural codecs a compelling choice for modern video streaming solutions.

In an effort to optimize video compression beyond traditional and current neural codec capabilities, we introduce an enhanced layered neural codec that utilizes both intra-frame and inter-frame redundancies. Our approach extends the layered neural codecs by incorporating motion vectors as a latent space, akin to the macroblock techniques used in H.264, to maximize inter-frame compression efficiency.

Algorithm 1 Neural Encoding and Compression using the video data inference pipeline.

```
1: Input: Video frames sequence F = \{f_1, f_2, \dots, f_n\}
2: Output: Compressed frames C = \{c_1, c_2, \dots, c_n\}
3: Initialize MobileNet Model M
4: Initialize Autoencoder Decoder D
5: Initialize Motion Vector Extractor V
6: procedure EXTRACTFEATURES(frame)
       features \leftarrow M(frame)
                                                             return features
8:
9: end procedure
10: procedure CompressFeatures(features)
       compressed \leftarrow D(features)
                                                                 12:
       return compressed
13: end procedure
14: procedure CalculateMotionVectors(frame_{current}, frame_{previous})
15:
       motion\_vectors \leftarrow V(frame_{current}, frame_{previous})
16:
       return\ motion\_vectors
17: end procedure
18: procedure STACKCOMPRESSION(current_compressed, motion_vectors)
19:
       return some compression algorithm using current_compressed and motion_vectors
20: end procedure
21: for i \leftarrow 1 to n do
       features_i \leftarrow \text{EXTRACTFEATURES}(f_i)
23:
       c_i \leftarrow \text{COMPRESSFEATURES}(features_i)
24:
       if i > 1 then
25:
          m_i \leftarrow \text{CALCULATEMOTIONVECTORS}(f_i, f_{i-1})
26:
          c_i \leftarrow \text{STACKCOMPRESSION}(c_i, m_i)
27:
       end if
28:
       C[i] \leftarrow c_i
29: end for
```

By exploiting the temporal correlations between consecutive frames, our codec can significantly reduce the required bitrate while maintaining high video quality.

The primary innovation lies in leveraging motion vectors to identify and encode only the changes between frames, rather than re-encoding entire frames. This technique, combined with the use of anchor frames—similar to keyframes in traditional codecs—allows the codec to understand and predict frame sequences more effectively, reducing redundancy and enhancing compression. Let F_t represent the frame at time t, and F_{t-1} be the anchor frame. The motion vector field M_t between F_t and F_{t-1} is computed to capture the displacement of pixels. Each frame F_t is divided into blocks $B_{t,i}$, where i indexes the block within the frame. The residual frame R_t for each frame is calculated as:

$$R_t = F_t - \operatorname{predict}(F_{t-1}, M_t)$$

where $\operatorname{predict}(\cdot)$ is a function that reconstructs F_t from F_{t-1} using the motion vector field M_t . This prediction involves translating the blocks of F_{t-1} according to M_t and serves as the predicted frame. The residual frame R_t , which contains only the differences not captured by the motion prediction, is then encoded using the layered neural network. Each layer of the neural network encodes progressively finer details of R_t . If $L_k(R_t)$ represents the k-th layer's encoding of the residual frame, the overall encoding of the frame can be expressed as:

$$E_t = \sum_{k=1}^K L_k(R_t)$$

where K is the total number of layers, and each L_k encodes different levels of detail or different regions of the frame, based on the motion information and the prediction error. Algorithm 1 shows the details of the layered neural codec.

We implement this neural codec using the FPGA in the CSDs. FPGAs are ideal for this application due to their parallel processing capabilities and the ability to handle multiple data streams concurrently.

Algorithm 2 Training Auto-Encoder with Motion Vectors and Stacked Compression while freezing the inference model.

```
1: Input: Set of training video sequences V
2: Initialize: MobileNet M
                                                                             ▶ Weights frozen
3: Initialize: Autoencoder A
                                                                                  ▶ Trainable
4: Initialize: Motion Vector Extractor V
5: procedure EXTRACTMOTIONVECTORS(frame_{current}, frame_{previous})
       motion\_vectors \leftarrow V(frame_{current}, frame_{previous})
       return motion vectors
8: end procedure
   procedure FORWARDPASS(video)
       previous\_features \leftarrow \texttt{null}
10:
       previous\_compressed \leftarrow null
11:
       for each frame frame in video do
12:
           features \leftarrow M(frame)
13:
                                                     compressed \leftarrow A.encode(features)
                                                                          14:
          if previous \ compressed \neq null then
15:
              motion\ vectors \leftarrow \text{ExtractMotionVectors}(frame, previous\ frame)
16:
              stacked\ input \leftarrow concatenate
17:
   (compressed, previous\_compressed, motion\_vectors)
              compressed \leftarrow A.reencode(stacked\ input)
18:
                                                                       19:
          end if
          reconstructed \leftarrow A.decode(compressed)
                                                                  ▷ Decompress to reconstruct
20:
          Calculate reconstruction loss between frame and reconstructed
21:
          previous frame \leftarrow frame
22:
          previous \ compressed \leftarrow compressed
23:
          previous\_features \leftarrow features
24:
25:
       end for
       Backpropagate loss and update weights of A only
26:
27: end procedure
28: while not converged do
29:
       for each video in V do FORWARDPASS(video)
30:
       end for
31: end while
```

The implementation of layered codecs involve the following components. **1. Motion Estimation:** Utilize dedicated hardware blocks for calculating motion vectors between consecutive frames. This step can leverage FPGA's DSP slices for fast cross-correlation or block matching algorithms. **2. Prediction and Residual Calculation:** Implement pipelined architectures for the predict(·) function and subsequent residual calculation to minimize latency. **3. Layered Encoding:** Each layer of the neural codec can be implemented using parallel processing units in FPGA, allowing simultaneous processing of different frame parts or different quality layers. **4.Data Flow Management:** Design efficient data paths to handle the high throughput of video data and intermediate results between the FPGA blocks, ensuring that bandwidth and memory access bottlenecks are minimized. By optimizing each component for FPGA execution and taking advantage of the hardware's ability to execute multiple operations in parallel, the proposed codec can achieve real-time performance even for high-resolution video streams. The use of FPGA not only accelerates the processing speed but also provides flexibility to adapt to various codec configurations.

<u>Training the Neural Codecs to Utilize Inference Pipeline:</u> The enhancement of our layered neural codec involves a joint training regimen that integrates the model used in inference pipeline, specifically MobileNet, as a static feature extractor within the compression framework. This strategy capitalizes on the robust, pre-trained features of MobileNet, which are frozen during training to ensure their integrity and to leverage their proven capability in capturing essential visual features. The extraction of motion vectors between frames further enriches the feature space by incorporating temporal dynamics essential for effective compression.

The codec's autoencoder component, which is trainable, is then tasked with compressing these enriched features. This setup not only streamlines the encoding process by utilizing high-quality

features but also significantly enhances compression efficiency by exploiting both intra-frame richness and inter-frame continuity. The training process is designed to optimize the autoencoder's ability to compress and decompress video sequences efficiently, without altering the pretrained feature extractor, thereby providing a stable, high-performance baseline for feature representation. The mathematical formulation for this training process is centered around minimizing the reconstruction loss, $L = \sum_{t=1}^N \|F_t - \hat{F}_t\|_2^2$; where F_t is the original frame at time t, and \hat{F}_t is the reconstructed frame, obtained by decoding the compressed representation that was encoded using features extracted via MobileNet and refined by the motion vector-informed autoencoder. The backpropagation is applied only to the layers of the autoencoder, ensuring that the feature extractor's parameters remain intact. This approach not only preserves the integrity of the visual features derived from MobileNet but also tailors the compression mechanism to be highly adaptive to the content-specific characteristics captured by these features.

4 Optimizing Lattice-Based Cryptography

Followed by the video compression, in this section, we discuss the quantum safe encryption technique. Among quantum safe encryption algorithms, Lattice-based cryptography (LBC) (Micciancio & Regev, 2009; Ajtai, 1996), grounded in hard lattice problems, offers robust resistance against quantum attacks. This quantum resilience is vital for protecting large data-sets on machine learning storage servers, particularly from 'store now, decrypt later' threats.

Algorithm 3 Lattice-Based Encryption Process

- 1: **procedure** LATTICE_BASED_ENCRYPT(message, public_key)
- 2: PM ← ConvertToPolynomial(message)
- 3: (PA, PE) ← GenerateRandomPolynomials()
- 4: $C1 \leftarrow PolynomialMultiply(PA, public key)$
- 5: $C2 \leftarrow PolynomialMultiply(PA, PM)$

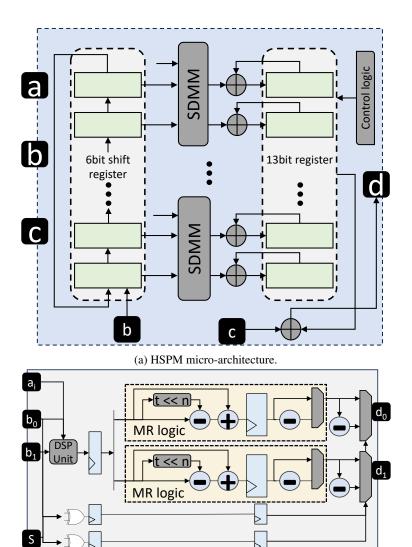
▷ Utilizing HSPM▷ Employing SDMM

- 6: **return** (C1, C2)
- 7: end procedure

At the heart of LBC lies the Ring-Learning with Errors (R-LWE) algorithm, which translates plaintext messages into polynomial representations and intertwines them with random polynomials using complex multiplications and additions, as delineated in Algorithm 3. Note that, within the LBC algorithm, certain components, specifically polynomial multiplications, exhibit similarities to operations performed in CNNs. These similarities open up the possibility of reusing hardware designed for CNN operations to accelerate LBC computations. The most common algorithms used fod the said operations are High-Speed Schoolbook Polynomial Multiplication (HSPM) and Pipelined Systolic Dimension Modular Multiplier (SDMM), and we propose to accelerate the same using the FPGAs in the CSDs. The data locality due to the SSD and the high throughput due to the FPGA could facilitate swift polynomial processing and refined modular multiplications using DSP slices, thereby accelerating the encryption process.

Designing HSPM Accelerator on CSD FPGA: The HSPM hardware is characterized by its fully parallelized design, incorporating 128 Multiply-Accumulate (MAC) units for handling polynomials of degree n=256. Each MAC unit within the HSPM architecture is capable of conducting two parallel modular multiplications. This is achieved through the use of a single Digital Signal Processing (DSP) block that operates on signed data representation. Consequently, these units are referred to as Signed Double Modular Multiplication (SDMM) units. The HSPM accelerator's architecture, as illustrated in Fig. 3a, comprises three pipelined stages. These stages include the data loading phase, the modular polynomial multiplication via the SDMM unit, and the final accumulation registers. Data is input into the HSPM in a serial-to-parallel conversion process, while the outputs, after undergoing addition operations, are retrieved serially through an address signal.

Central to the Ring-Learning with Errors (R-LWE) based Public Key Encryption (PKE) is the equation $d=a\cdot b+c$. In this context, the operand a can represent polynomials such as p, a, or c_1 from Algorithm 1, while b corresponds to e_1 , r_2 , and c to e_2 , e_3 , c_2 . During the data loading phase, the 256 coefficients of polynomial b are input serially into a 6-bit shift register. Simultaneously, the first coefficient a_0 of polynomial a is fed in parallel to all 128 SDMM units. Post-loading, each SDMM



(b) Pipeline SDMM Structure micro-architecture.

Figure 3: Microarchitectural designs of HSPM and SDMM: Hardware modules for polynomial multiplication in LBC.

unit commences the modular multiplication of each coefficient a_i with the entirety of b's coefficients in parallel. This process repeats for all coefficients a_i . The resultant outputs from each SDMM calculation are accumulated every cycle. The final polynomial product p is sequentially read out by the address signal addr_{ab} , combined with the coefficient of c, thereby producing the final output d as $d=a\cdot b+c$. This output is then transmitted serially over n clock cycles.

Designing SDMM Hardware on CSD FPGA: The SDMM hardware is innovatively designed to perform two modular multiplications per DSP Slice. This is achieved through the implementation of signed Gaussian sampling (Liu et al., 2019), for the error-vector e_1 and the secret-key r_2 . In this signed representation, the Gaussian distribution ranges from [0, ks) to [q - ks, 0), where k takes integer values $1, 2, 3, \ldots$, and is symmetrically distributed around m = 0. Consequently, there is no data in the range [0, q - 1), allowing the maximum value of the 13-bit samples to be represented efficiently by a 6-bit signed number. The modular multiplication utilizing these signed numbers is formulated as follows:

$$b \otimes a \mod q = q - [(b - a) \mod q] \tag{1}$$

Here, the most significant bit (MSB) signed-bit dictates the subtraction of the result from the modulus q, as illustrated in Fig. 3b. A single DSP unit includes a multiplier, with the lowest 18 bits of the

DSP output representing the first product d_0 and the highest 18 bits indicating the second product d_1 , corresponding to b_0 and b_1 respectively, as depicted in Fig. 3b.

Following the multiplication, the two 18-bit products undergo separate modular reductions. A Modular Reduction (MR) circuitry based on approximation (Kundi et al., 2020). Our MR unit achieves consumes $\approx 82\%$ less hardware compared to classical implementation. This process is constant time, requiring only a single subtraction of q. The MR hardware, highlighted in Fig. 3b, consists of a single shift block, a subtractor, and a adder, consuming much less hardware then the state-of-the-art (Fan et al., 2018). The MR, in conjunction with the multiplier, delivers the output within 2 clock cycles. The SDMM also offers a simpler controller design for multiplication and in-place product term reduction after vector-vector multiplications. Notably, the SDMM can be easily reconfigured to support any modulus.

5 Implementation and Evaluation

To implement and evaluate $Salient\ Store$, we chose two different platforms -1) Using a server-grade system with two Xilinx-Samsung computational storage drive (AMD, b), and 2) Using amazon web services (AWS) F1 instances, which have AMD Alveo FPGA which can work as the compute capable of peer-to-peer communication and thereby enabling a computational storage platform. We use Vitis 2022.2 along with Xilinx Runtime Library (AMD & Xilinx) for programming the FPGAs in both computational storage drives and the Alveo card. The configuration of the server with Xilinx CSD has a 12-core Xeon bronze CPU with 128GB memory, 2×3.84 TB CSDs, and 2×2 TB SSDs. Similarly, the AWS server has 24 cores, with 192GB memory, one Alveo FPGA and 2TB SSDs.

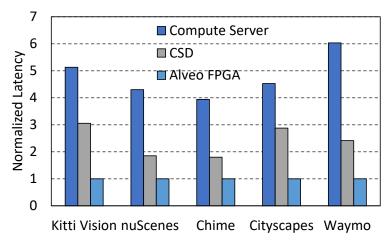
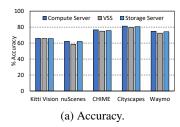
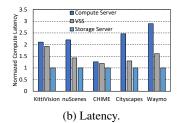


Figure 4: Latency analysis of *Salient Store* on the commercial Xilinx CSD on a workstation class machine (lower is better). Compute server indicates a software only classical storage solution without CSDs.

Data Location	kernel Execution	SpeedUp
CSD1	CPU	1
CSD1	CSD1	3.9
CSD1 (0.1X), CSD2(0.9X)	CSD1 (0.1X), CSD2(0.9X)	4.46
CSD1 (0.3X), CSD2(0.7X)	CSD1 (0.3X), CSD2(0.7X)	5.608
CSD1 (0.4X), CSD2(0.6X)	CSD1 (0.4X), CSD2(0.6X)	6.67
CSD1 (0.5X), CSD2(0.5X)	CSD1 (0.5X), CSD2(0.5X)	7.7

Table 2: Effect of Data distribution on compute speed-up.





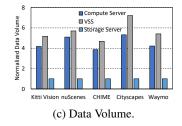


Figure 5: Performance of *Salient Store* on larger compute and storage nodes. This experiment to mimics a consolidated edge server catering to many video streams. Compute server indicates a software only classical storage solution without CSDs.

5.1 Evaluation of Encoding, Scaling and Accuracy

In evaluating the Salient Store storage system, particularly for continuous learning scenarios video analytics applications, we selected data-sets that are not only dense but also necessitate continuous learning due to their dynamic nature. Autonomous driving and urban mobility applications, generating over 400TB of data annually (Wright"; "premioinc"; Bhardwaj et al., 2022), predominantly comprise video and 3D point cloud data, making them ideal for our evaluation. The Cityscapes data-set (Cordts et al., 2015) is a comprehensive collection of urban street scenes from 50 different cities, providing a rich source of annotated video data for semantic urban scene understanding. This data-set is particularly suited for evaluating the performance of Salient Store in dense, urban environments. Another video data-set, the Waymo Open Data-set (Sun et al., 2020), is one of the largest and most diverse data-sets for autonomous driving. It offers high-resolution sensor data, including LIDAR and camera recordings, across a variety of urban and suburban landscapes. This data-set's volume and diversity make it an excellent benchmark for assessing Salient Store 's capabilities in handling large-scale, real-world data. For 3D point cloud data, we selected the KITTI Vision Benchmark Suite (Geiger et al., 2012), a fundamental data-set in autonomous driving research. It includes a variety of data types and tasks, providing a real-world urban driving context that is essential for testing Salient Store 's performance in processing and managing 3D spatial data. Additionally, the nuScenes data-set (Caesar et al., 2020) by Aptiv, with its comprehensive sensor data and annotations covering diverse driving scenes, is instrumental in evaluating Salient Store 's efficiency in multi-modal data processing typical in urban mobility scenarios. Beyond the visionbased data, we also incorporated the Chime Audio data-set (Foster et al., 2015) into our evaluation. This data-set, consisting of audio recordings, offers a different modality to test the versatility of Salient Store in handling various types of continuous learning data beyond visual inputs. In our comparative analysis, we employ VSS: A Video Storage System (Haynes et al., 2021) as a "baseline", given its innovative approach in optimizing video data management. VSS excels in decoupling high-level video operations from storage and retrieval processes, efficiently organizing data on disk, enhancing caching mechanisms, and reducing redundancies in multi-camera setups (Haynes et al., 2021). This system has demonstrated significant improvements in VDBMS read performance, up to 54%, and a reduction in storage costs by up to 45% (Haynes et al., 2021). VSS's emphasis on video data optimization makes it a pertinent benchmark for assessing the Salient Store storage system's capabilities in managing large-scale machine learning data-sets.

We first implemented $Salient\ Store$ on a workstation-class machine with two Xilinx CSDs - which colosely mimics the classical edge server setup. Fig. 4 shows the latency while using the storage server and CSD for performing data writes, normalized to the latency of doing the same using the state of the art Alveo FPGAs. $Salient\ Store$ on CSDs perform $\approx 1.99 \times$ better than the implementation on classical storage systems. To further mimic realwold scenarios of multiple cameras sending multiple streams with various stream rate, we distributed video data into across two CSDs in different ratio, as shown in TABLE 2. Trivially, a load-balanced scenario outperforms any of the biased data distribution scenario. However, it is evident that any compute offloaded to any of the CSDs in these scenarios offers benefits in terms of data processing latency.

To underscore the impact of *Salient Store* at a much larger scale, where one can assume a consolidated edge server catering towards multiple streams as depicted in Ekya (Bhardwaj et al., 2022), we utilized an AWS EC2 F1 instance with Alveo FPGAs, along with EC2 P4 instance with Al00 GPUs to implement the continuous learning end-to-end. As shown in Fig. 5a *Salient Store*

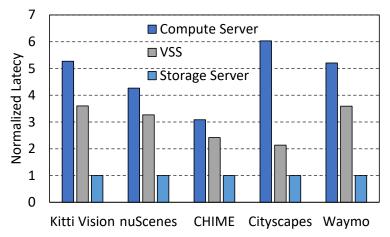


Figure 6: Impact of scaling to multiple storage nodes.

keeps up with the accuracy of the traditional compression system thanks to the robust layered coding algorithm. Additionally, as shown in Fig. 5b, Salient Store has $4.49\times$ less latency than VSS where as it shows about a $6.18\times$ speedup compared to the classical storage server. Furthermore, as depicted in Fig. 5c, SLAT reduces the data communication volume up to $\approx 5.63\times$ compared to the classical approaches, thereby saving bandwidth (reduces bandwidth by $\approx 36\%$, public cloud has strict regulations on measuring the actual bandwidth, and hence we report the approximate result from the traffic pattern) and energy.

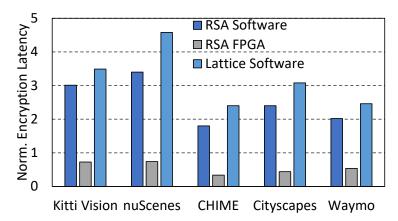


Figure 7: Proposed encryption vs the state-of-the-art normalized to the FPGA implementation of the lattice based encryption.

Since majority of the storage systems are not limited to one storage server, but are spread across multiple servers, we scaled <code>Salient Store</code> by deploying it in a distributed fashion. We used $5\times$ AWS EC2 F1 instances with Alveo FPGAs as storage nodes, along with one EC2 P4 instance with A100 GPUs as the compute server. We do not observe much change in the data volume. However, due to complex interaction between multiple storage nodes, especially where some of the data needed to arrive at different nodes via network, in Fig. 6, we observe a significant change in the latency compared to a single storage node. Although, a multi-node setup provides more parallelism, the speedup is sub-linear, and we observe $\approx 3\times$ and $\approx 4.77\times$ speedup against VSS and a classical storage server, respectively.

5.2 Evaluation of Video Data Recovery and Quality:

To ensure proper video quality upon recovery, we perform a peak signal-to-noise ratio (PSNR) study of *Salient Store* with the classical encoding mechanisms H264 (ITU-T, 2019a) and H265 (HEVC) (ITU-T, 2019b). Experiments on waymo (Sun et al., 2020) dataset shows the PSNR of *Salient Store* compared to the classical H264 and HEVC encoding pipeline in Fig. 8. While

Salient Store is consistently performing better than H264, HEVC thanks to it's novel approach of fine grained computation at times outperforms our approach. A similar trend is observed in other datasets. With complex and high dimensional video data, HEVC computation complexity increases exponentially, and is more pronounced because of lack of hardware support. However, it is noteworthy that at higher bitrates, $Salient\ Store$ consistently achieves high quality recovery with PSNR reaching $\approx 47 \text{dB}$. Furthermore, as we can see in Fig. 9, HEVC takes significantly more latency compared to $Salient\ Store$ and therefore not entirely suitable for high frame-rate applications with resource constraints.

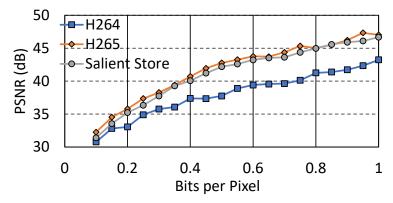


Figure 8: Compression and Recovery efficiency.

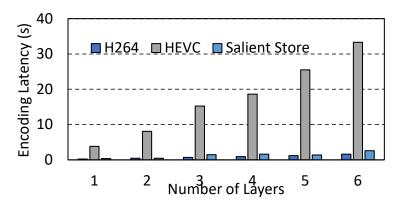


Figure 9: Encoding Latency using layered coding.

5.3 Evaluation of Lattice Based Encryption

One of the major contribution of <code>Salient Store</code> is accelerating the lattice-based encryption with the help of FPGAs on the storage nodes. Fig. 7 shows the comparison of our FPGA-accelerated lattice-based encryption against other state-of-the-art-techniques. RSA, the most popular encryption algorithm, when implemented using FPGA, outperforms our proposed hardware solution. However, our proposed solution offers $\approx 3.2\times$ speedup compared to its software counterpart and a $\approx 2.5\times$ speedup compared to the software based RSA algorithm. While we do agree that the lattice-based encryption has more overheads compared to the RSA algorithm, the benefit of being "quantum-safe" outweighs the minimal cost incurred on encryption.

Fig. 10 shows the impact of scaling Salient Store with respect to a single node system, i.e., a single storage server. As the number of storage servers increases, the network contention and data orchestration challenges exponentially increase. Furthermore, each server performs more remote accesses, increasing the contention even more. This leads to an exponential growth in latency with the increase in the number of storage servers used per application. It is recommended to contain most of the data belonging to the same application in the same storage server to minimize remove disk accesses over network.

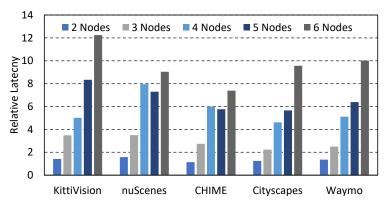


Figure 10: Change of data movement latency with respect to the number of storage servers.

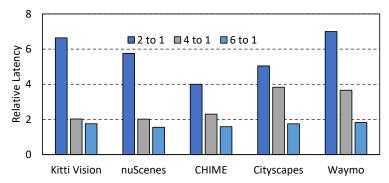


Figure 11: Impact of increasing the number of CSDs in the system (the baseline system has a SSD-to-CSD ratio 8 to 1).

Similarly, Fig. 11 shows the impact of increasing the number of CSDs in a storage server. With increasing the number of CSDs per SSD (or other storage element) does show significant improvement because of increase in parallelism. However, a typical CSD is $\approx 15\times$ expensive than a standard SSD and $\approx 25\times$ expensive than a classical HDD. Integrating more number of CSDs into the standard storage server will significantly increase the cost of the server. Moreover, in large-scale systems where failure is common, replacing failed CSDs would further increase the cost. From our evaluation, we found an 8:1 ratio of SSD to CSD (capacity ratio) provides the best possible cost-to-acceleration benefit.

6 Conclusions

In this paper, we have explored the critical role of storage systems in the context of continuous learning video analytics edge server, emphasizing in particular the transformative potential of Computational Storage Devices (CSDs) in this domain. Our proposal, $Salient\ Store$, highlights the need for a paradigm shift in storage architecture to accommodate the dynamic and computationally intensive nature of modern ML applications. By reducing unnecessary data movement and enabling near-data processing, CSDs enhance the efficiency, performance, and sustainability of storage servers. $Salient\ Store$, with its intelligent data orchestration and acceleration, can provide up to $6.18\times$ latency and $6.13\times$ data movement reduction, compared to classical systems. This is particularly evident in continuous learning scenarios prevalent in applications such as autonomous driving and urban mobility, where the volume and complexity of data are immense.

Looking ahead, the role of storage systems is poised to become even more pivotal as ML applications continue to evolve and generate larger, more complex datasets. The ongoing innovation in the design and optimization of CSDs will be crucial in meeting these challenges. In summary, our research presents a comprehensive vision for the future of storage systems in ML, where computational storage

devices play a key role in advancing the, performance, efficiency, and capabilities of storage servers, thereby contributing significantly to the broader field of ML.

References

- Achieving Compliant Data Residency and Security with Azure. Achieving Compliant Data Residency and Security with Azure. https://azure.microsoft.com/mediahandler/files/resourcefiles/achieving-compliant-data-residency-and-security-with-azure/Achieving_Compliant_Data_Residency_and_Security_with_Azure.pdf.
- Miklós Ajtai. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 99–108, 1996.
- AMD. Vitis unified software platform. https://www.xilinx.com/products/design-tools/vitis/vitis-platform.html, a. (Accessed on 11/13/2023).
- AMD. Smartssd computational storage drive. https://www.xilinx.com/applications/data-center/computational-storage/smartssd.html, b. (Accessed on 07/13/2023).
- AMD and Xilinx. Xilinx runtime library (xrt). https://www.xilinx.com/products/design-tools/vitis/xrt.html. (Accessed on 11/20/2023).
- Guilherme H. Apostolo, Pablo Bauszat, Vinod Nigade, Henri E. Bal, and Lin Wang. Live video analytics as a service. In *Proceedings of the 2nd European Workshop on Machine Learning and Systems*, EuroMLSys '22, pp. 37–44, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392549. doi: 10.1145/3517207.3526973. URL https://doi.org/10.1145/3517207.3526973.
- David Arthur and Sergei Vassilvitskii. K-means++ the advantages of careful seeding. In *Proceedings* of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, pp. 1027–1035, 2007.
- Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *arXiv preprint arXiv:1203.6402*, 2012.
- Antonio Barbalace and Jaeyoung Do. Computational storage: Where are we today? In CIDR, 2021.
- Romil Bhardwaj, Zhengxu Xia, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Nikolaos Karianakis, Kevin Hsieh, Paramvir Bahl, and Ion Stoica. Ekya: Continuous learning of video analytics models on edge compute servers. In 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22), pp. 119–135, 2022.
- Ilker Bozcan and Erdal Kayacan. Au-air: A multi-modal unmanned aerial vehicle dataset for low altitude traffic surveillance, 2020. URL https://arxiv.org/abs/2001.11737.
- M. Bramberger, J. Brunner, B. Rinner, and H. Schwabach. Real-time video analysis on an embedded smart camera for traffic surveillance. In *Proceedings. RTAS 2004. 10th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2004., pp. 174–181, 2004. doi: 10.1109/RTTAS.2004.1317262.
- Barry Brown, Mathias Broth, and Erik Vinkhuyzen. The halting problem: Video analysis of self-driving cars in traffic. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394215. doi: 10.1145/3544548.3581045. URL https://doi.org/10.1145/3544548.3581045.
- Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11621–11631, 2020.
- Jie Cao, Zhiang Wu, Junjie Wu, and Wenjie Liu. Towards information-theoretic k-means clustering for image indexing. *Signal Processing*, 93(7):2026–2037, 2013.

- Wei Cao, Yang Liu, Zhushi Cheng, Ning Zheng, Wei Li, Wenjie Wu, Linqiang Ouyang, Peng Wang, Yijing Wang, Ray Kuan, et al. {POLARDB} meets computational storage: Efficiently support analytical workloads in {Cloud-Native} relational database. In 18th USENIX conference on file and storage technologies (FAST 20), pp. 29–41, 2020.
- Keith Chapman, Mehdi Nik, Behnam Robatmili, Shahrzad Mirkhani, and Maysam Lavasani. Computational storage for big data analytics. In *Proceedings of 10th International Workshop on Accelerating Analytics and Data Management Systems (ADMS'19)*, 2019.
- Jianyu Chen, Maurice Daverveldt, and Zaid Al-Ars. Fpga acceleration of zstd compression algorithm. In 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 188–191. IEEE, 2021.
- Tong Chen, Haojie Liu, Qiu Shen, Tao Yue, Xun Cao, and Zhan Ma. Deepcoder: A deep neural network based video compression. In 2017 IEEE Visual Communications and Image Processing (VCIP), pp. 1–4. IEEE, 2017.
- Yihua Cheng, Ziyi Zhang, Hanchen Li, Anton Arapin, Yue Zhang, Qizheng Zhang, Yuhan Liu, Kuntai Du, Xu Zhang, Francis Y Yan, et al. {GRACE}:{Loss-Resilient}{Real-Time} video through neural codecs. In 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24), pp. 509–531, 2024.
- Yann Collet and Murray Kucherawy. Zstandard compression and the application/zstd media type. Technical report, 2018.
- Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Scharwächter, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset. In CVPR Workshop on the Future of Datasets in Vision, volume 2. sn, 2015.
- Intel Corporation. Smart city technologies and intelligent transportation systems are helping cities absorb growing populations, overcome congestion, and create sustainable futures. https://www.intel.com/content/www/us/en/transportation/urban-mobility.html. (Accessed on 11/21/2022).
- Custom On-Device ML Models with Learn2Compress. Custom on-device ml models with learn2compress. https://ai.googleblog.com/2018/05/custom-on-device-ml-models.html. (Accessed on 11/21/2022).
- Mallesham Dasari, Kumara Kahatapitiya, Samir R Das, Aruna Balasubramanian, and Dimitris Samaras. Swift: Adaptive video streaming with layered neural codecs. In 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22), pp. 103–118, 2022a.
- Mallesham Dasari, Kumara Kahatapitiya, Samir R. Das, Aruna Balasubramanian, and Dimitris Samaras. Swift: Adaptive video streaming with layered neural codecs. In 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22), pp. 103–118, Renton, WA, April 2022b. USENIX Association. ISBN 978-1-939133-27-4. URL https://www.usenix.org/conference/nsdi22/presentation/dasari.
- Maureen Daum, Brandon Haynes, Dong He, Amrita Mazumdar, and Magdalena Balazinska. Tasm: A tile-based storage manager for video analytics. In 2021 IEEE 37th International Conference on Data Engineering (ICDE), pp. 1775–1786. IEEE, 2021.
- Jaeyoung Do, Yang-Suk Kee, Jignesh M Patel, Chanik Park, Kwanghyun Park, and David J DeWitt. Query processing on smart ssds: Opportunities and challenges. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 1221–1230, 2013.
- Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *arXiv preprint arXiv:2401.08281*, 2024.
- Sourav Dutta and Chinwe Ekenna. Air-to-ground surveillance using predictive pursuit. In 2019 International Conference on Robotics and Automation (ICRA), pp. 8234–8240. IEEE Press, 2019. doi: 10.1109/ICRA.2019.8794073. URL https://doi.org/10.1109/ICRA.2019.8794073.

- Eideticom and Los Alamos National Laboratory. Line-rate compression on lustre/zfs-based parallel filesystems using noload computational storage processor. https://www.eideticom.com/media/attachments/2020/06/03/noload-compression-zfs.pdf. (Accessed on 11/13/2023).
- Lizhou Fan, Zhanyuan Yin, Huizi Yu, and Anne J Gilliland. Using machine learning to enhance archival processing of social media archives. *Journal on Computing and Cultural Heritage* (*JOCCH*), 15(3):1–23, 2022.
- Sailong Fan, Weiqiang Liu, James Howe, Ayesha Khalid, and Maire O'Neill. Lightweight hardware implementation of r-lwe lattice-based cryptography. In 2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), pp. 403–406. IEEE, 2018.
- Shaoheng Fang, Zi Wang, Yiqi Zhong, Junhao Ge, Siheng Chen, and Yanfeng Wang. Tbp-former: Learning temporal bird's-eye-view pyramid for joint perception and prediction in vision-centric autonomous driving, 2023. URL https://arxiv.org/abs/2303.09998.
- Manuel J Fonseca and Joaquim A Jorge. Indexing high-dimensional data for content-based retrieval in large databases. In *Eighth International Conference on Database Systems for Advanced Applications*, 2003.(DASFAA 2003). Proceedings., pp. 267–274. IEEE, 2003.
- Peter Foster, Siddharth Sigtia, Sacha Krstulovic, Jon Barker, and Mark D Plumbley. Chime-home: A dataset for sound source recognition in a domestic environment. In 2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), pp. 1–5. IEEE, 2015.
- Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In 2012 IEEE conference on computer vision and pattern recognition, pp. 3354–3361. IEEE, 2012.
- Maksim Godovykh, Carissa Baker, and Alan Fyall. Vr in tourism: A new call for virtual tourism experience amid and after the covid-19 pandemic. *Tourism and Hospitality*, 3(1):265–275, 2022.
- Philipp M. Grulich and Faisal Nawab. Collaborative edge and cloud neural networks for real-time video processing. *Proc. VLDB Endow.*, 11(12):2046–2049, aug 2018. ISSN 2150-8097. doi: 10.14778/3229863.3236256. URL https://doi.org/10.14778/3229863.3236256.
- Brandon Haynes, Maureen Daum, Dong He, Amrita Mazumdar, Magdalena Balazinska, Alvin Cheung, and Luis Ceze. Vss: A storage system for video analytics. In *Proceedings of the 2021 International Conference on Management of Data*, pp. 685–696, 2021.
- Qunying Huang and Chen Xu. A data-driven framework for archiving and exploring social media data. *Annals of GIS*, 20(4):265–277, 2014.
- Zhiyu Huang, Haochen Liu, and Chen Lv. Gameformer: Game-theoretic modeling and learning of transformer-based interactive prediction and planning for autonomous driving, 2023. URL https://arxiv.org/abs/2303.05760.
- ITU-T. Advanced video coding for generic audiovisual services. International Telecommunication Union, June 2019a. URL https://www.itu.int/rec/T-REC-H.264. ITU-T Recommendation H.264.
- ITU-T. High efficiency video coding. International Telecommunication Union, June 2019b. URL https://www.itu.int/rec/T-REC-H.265. ITU-T Recommendation H.265.
- Myoungsoo Jung. Hello bytes, bye blocks: Pcie storage meets compute express link for memory expansion (cxl-ssd). In *Proceedings of the 14th ACM Workshop on Hot Topics in Storage and File Systems*, pp. 45–51, 2022.
- Hwajung Kim, Heon Y Yeom, and Hanul Sung. Understanding the performance characteristics of computational storage drives: A case study with smartssd. *Electronics*, 10(21):2617, 2021.
- Yoonsung Kim, Changhun Oh, Jinwoo Hwang, Wonung Kim, Seongryong Oh, Yubin Lee, Hardik Sharma, Amir Yazdanbakhsh, and Jongse Park. Dacapo: Accelerating continuous learning in autonomous systems for video analytics. *arXiv preprint arXiv:2403.14353*, 2024.

- Terhi Korkiakangas. Challenges in archiving and sharing video data: Considering moral, pragmatic and substantial arguments. *Journal of Research Practice*, 10(1), 2014.
- Dur E Shahwar Kundi, Song Bian, Ayesha Khalid, Chenghua Wang, Máire O'Neill, and Weiqiang Liu. Axmm: Area and power efficient approximate modular multiplier for r-lwe cryptosystem. In 2020 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5. IEEE, 2020.
- Los Alamos National Laboratory. Los alamos national laboratory and sk hynix to demonstrate first-of-a-kind ordered key-value store computational storage device. https://discover.lanl.gov/news/0728-storage-device/. (Accessed on 11/13/2023).
- Shiju Li, Kevin Tang, Jin Lim, Chul-Ho Lee, and Jongryool Kim. Computational storage for an energy-efficient deep neural network training system. In *European Conference on Parallel Processing*, pp. 304–319. Springer, 2023.
- Weiqiang Liu, Sailong Fan, Ayesha Khalid, Ciara Rafferty, and Máire O'Neill. Optimized schoolbook polynomial multiplication for compact lattice-based cryptography on fpga. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(10):2459–2463, 2019.
- Corne Lukken and Animesh Trivedi. Past, present and future of computational storage: A survey. *arXiv* preprint arXiv:2112.09691, 2021.
- Siwei Ma, Xinfeng Zhang, Chuanmin Jia, Zhenghui Zhao, Shiqi Wang, and Shanshe Wang. Image and video compression with neural networks: A review. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(6):1683–1698, 2019.
- Vikram Sharma Mailthody, Zaid Qureshi, Weixin Liang, Ziyan Feng, Simon Garcia De Gonzalo, Youjie Li, Hubertus Franke, Jinjun Xiong, Jian Huang, and Wen-mei Hwu. Deepstore: In-storage acceleration for intelligent queries. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 224–238, 2019.
- Michael Mesnier. Intel labs showcases multi-vendor, computational storage platform. https://community.intel.com/t5/Blogs/Tech-Innovation/Data-Center/Intel-Labs-Showcases-Multi-Vendor-Computational-Storage-Platform/post/1404651, August 2022. (Accessed on 11/13/2023).
- Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post-quantum cryptography*, pp. 147–191. Springer, 2009.
- Evgeny Milanov. The rsa algorithm. RSA laboratories, pp. 1–11, 2009.
- Cyan Subhra Mishra, Jack Sampson, Mahmut Taylan Kandemir, Vijaykrishnan Narayanan, and Chita R Das. Usas: A sustainable continuous-learning framework for edge servers. In 2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pp. 891–907. IEEE, 2024.
- National Cybersecurity Center of Excellence (NCCoE). Post-quantum cryptography migration: Nist sp 1800-38b preliminary draft. Technical report, National Institute of Standards and Technology (NIST), 2023. URL https://www.nccoe.nist.gov/sites/default/files/2023-12/pqc-migration-nist-sp-1800-38b-preliminary-draft.pdf. Accessed: 2024-08-01.
- National Institute of Standards and Technology (NIST). Post-quantum cryptography. National Institute of Standards and Technology, 2024. URL https://csrc.nist.gov/projects/post-quantum-cryptography. Accessed: 2024-08-01.
- Samsung Newsroom. Samsung electronics develops second-generation smartssd computational storage drive with upgraded processing functionality. https://bit.ly/3PXwecP. (Accessed on 11/13/2023).
- NVIDIA Corporation. Nvidia rtx dlss. https://developer.nvidia.com/rtx/dlss, 2024. Accessed: 2024-08-02.
- James Jie Pan, Jianguo Wang, and Guoliang Li. Vector database management techniques and systems. In *Companion of the 2024 International Conference on Management of Data*, pp. 597–604, 2024.

- Inhyuk Park, Qing Zheng, Dominic Manno, Soonyeal Yang, Jason Lee, David Bonnie, Bradley Settlemyer, Youngjae Kim, Woosuk Chung, and Gary Grider. Kv-csd: A hardware-accelerated key-value store for data-intensive applications. *dimensions*, 30:33.
- Lorenzo Pichierri, Guido Carnevale, Lorenzo Sforni, Andrea Testa, and Giuseppe Notarstefano. A distributed online optimization strategy for cooperative robotic surveillance. In 2023 IEEE International Conference on Robotics and Automation (ICRA), pp. 5537–5543, 2023. doi: 10.1109/ICRA48891.2023.10160700.
- Roberto Pierdicca, Michele Sasso, Flavio Tonetto, Francesca Bonelli, Andrea Felicetti, and Marina Paolanti. Immersive insights: virtual tour analytics system for understanding visitor behavior. In Augmented Reality, Virtual Reality, and Computer Graphics: 8th International Conference, AVR 2021, Virtual Event, September 7–10, 2021, Proceedings 8, pp. 135–155. Springer, 2021.
- "premioinc". How much data do autonomous vehicles generate? https://premioinc.com. (Accessed on 11/13/2023).
- Abhishek Rawat, Kartik Sehgal, Amartya Tiwari, Abhishek Sharma, and Ashish Joshi. A novel accelerated implementation of rsa using parallel processing. *Journal of Discrete Mathematical Sciences and Cryptography*, 22(2):309–322, 2019.
- Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.
- Sahand Salamat, Armin Haj Aboutalebi, Behnam Khaleghi, Joo Hwan Lee, Yang Seok Ki, and Tajana Rosing. Nascent: Near-storage acceleration of database sort on smartssd. In *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 262–272, 2021.
- Sahand Salamat, Hui Zhang, Yang Seok Ki, and Tajana Rosing. Nascent2: Generic near-storage sort accelerator for data analytics on smartssd. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 15(2):1–29, 2022.
- ScaleFlux. Csd 2000. https://scaleflux.com/products/csd-2000/, a. (Accessed on 11/13/2023).
- ScaleFlux. Csd 3000. https://scaleflux.com/products/csd-3000/, b. (Accessed on 11/13/2023).
- Debendra Das Sharma. Compute express link (cxl): Enabling heterogeneous data-centric computing with heterogeneous memory hierarchy. *IEEE Micro*, 43(2):99–109, 2022a.
- Debendra Das Sharma. Compute express link®: An open industry-standard interconnect enabling heterogeneous data-centric computing. In 2022 IEEE Symposium on High-Performance Interconnects (HOTI), pp. 5–12. IEEE, 2022b.
- Heng Tao Shen, Beng Chin Ooi, and Xiaofang Zhou. Towards effective indexing for very large video sequence database. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pp. 730–741, 2005.
- Vibhaalakshmi Sivaraman, Pantea Karimi, Vedantha Venkatapathy, Mehrdad Khani, Sadjad Fouladi, Mohammad Alizadeh, Frédo Durand, and Vivienne Sze. Gemino: Practical and robust neural compression for video conferencing. In 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24), pp. 569–590, 2024.
- Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, and Benjamin et al. Caine. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2446–2454, 2020.
- Yao Tian, Xi Zhao, and Xiaofang Zhou. Db-lsh 2.0: Locality-sensitive hashing with query-based dynamic bucketing. *IEEE Transactions on Knowledge and Data Engineering*, 2023.

- Mahdi Torabzadehkashi, Ali Heydarigorji, Siavash Rezaei, Hosein Bobarshad, Vladimir Alves, and Nader Bagherzadeh. Accelerating hpc applications using computational storage devices. In 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 1878–1885, 2019a. doi: 10.1109/HPCC/SmartCity/DSS. 2019.00259.
- Mahdi Torabzadehkashi, Siavash Rezaei, Ali HeydariGorji, Hosein Bobarshad, Vladimir Alves, and Nader Bagherzadeh. Computational storage: an efficient and scalable platform for big data and hpc applications. *Journal of Big Data*, 6:1–29, 2019b.
- Min-Han Tsai, Nalini Venkatasubramanian, and Cheng-Hsin Hsu. Analytics-aware storage of surveillance videos: Implementation and optimization. In 2020 IEEE International Conference on Smart Computing (SMARTCOMP), pp. 25–32. IEEE, 2020.
- Urban Traffic Dataset. https://github.com/edge-video-services/ekya#urban-traffic-dataset.
- Shaohua Wan, Zonghua Gu, and Qiang Ni. Cognitive computing and wireless communications on the edge for healthcare service robots. *Computer Communications*, 149:99–106, 2020. ISSN 0140-3664. doi: https://doi.org/10.1016/j.comcom.2019.10.012. URL https://www.sciencedirect.com/science/article/pii/S0140366419307960.
- "Simon Wright". Autonomous cars generate more than 300 tb of data per year. https://www.tuxera.com/blog/autonomous-cars-300-tb-of-data-per-year/. (Accessed on 11/13/2023).
- www.dlapiperdataprotection.com. Sweden data collection & processing. https://www.dlapiperdataprotection.com/index.html?t=collection-and-processing&c=SE. (Accessed on 11/21/2022).
- Tiantu Xu, Luis Materon Botelho, and Felix Xiaozhu Lin. Vstore: A data store for analytics on large videos. In *Proceedings of the Fourteenth EuroSys Conference 2019*, pp. 1–17, 2019.
- Yang Yang, Zhi Guan, Huiping Sun, and Zhong Chen. Accelerating rsa with fine-grained parallelism using gpu. In *Information Security Practice and Experience: 11th International Conference, ISPEC 2015, Beijing, China, May 5-8, 2015, Proceedings*, pp. 454–468. Springer, 2015.
- Ziyu Ying, Shulin Zhao, Sandeepa Bhuyan, Cyan Subhra Mishra, Mahmut T. Kandemir, and Chita R. Das. Pushing point cloud compression to the edge. In 2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 282–299, 2022a. doi: 10.1109/MICRO56248. 2022.00031.
- Ziyu Ying, Shulin Zhao, Haibo Zhang, Cyan Subhra Mishra, Sandeepa Bhuyan, Mahmut T. Kandemir, Anand Sivasubramaniam, and Chita R. Das. Exploiting frame similarity for efficient inference on edge devices. In 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS), pp. 1073–1084, 2022b. doi: 10.1109/ICDCS54860.2022.00107.
- Hang Yue, Laurence R Rilett, and Peter Z Revesz. Spatio-temporal traffic video data archiving and retrieval system. *GeoInformatica*, 20:59–94, 2016.
- Haibo Zhang, Prasanna Venkatesh Rengasamy, Shulin Zhao, Nachiappan Chidambaram Nachiappan, Anand Sivasubramaniam, Mahmut T. Kandemir, Ravi Iyer, and Chita R. Das. Race-to-sleep + content caching + display caching: a recipe for energy-efficient video streaming on handhelds. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, pp. 517–531, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349529. doi: 10.1145/3123939.3123948. URL https://doi.org/10.1145/3123939.3123948.
- Shulin Zhao, Haibo Zhang, Sandeepa Bhuyan, Cyan Subhra Mishra, Ziyu Ying, Mahmut T. Kandemir, Anand Sivasubramaniam, and Chita R. Das. Déjà view: Spatio-temporal compute reuse for energy-efficient 360° vr video streaming. In 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), pp. 241–253, 2020. doi: 10.1109/ISCA45697.2020.00030.

Shulin Zhao, Haibo Zhang, Cyan Subhra Mishra, Sandeepa Bhuyan, Ziyu Ying, Mahmut Taylan Kandemir, Anand Sivasubramaniam, and Chita Das. Holoar: On-the-fly optimization of 3d holographic processing for augmented reality. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '21, pp. 494–506, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450385572. doi: 10.1145/3466752.3480056. URL https://doi.org/10.1145/3466752.3480056.

Jingjie Zhu, Mingming Cheng, and Ying Wang. Viewer in-consumption engagement in proenvironmental tourism videos: A video analytics approach. *Journal of Travel Research*, pp. 00472875231219634, 2024.