

Enhanced Real-Time Analytics through Edge Computing and Efficient Data Mining for Optimised Latency in IoT Data Streams

Dr. S. M. Shamsheer Daula¹, Prof. Dr. Martin Margala², Prof. Dr. Mohammed Ali Hussain³ and Prof. Dr. Prasun Chakrabarti⁴

¹Postdoctoral Fellow at University of Louisiana and Associate Professor, Department of ECE, G Pulla Reddy Engineering College (A), Kurnool, A.P, India,

²Director of School of Computing and Informatics, University of Louisiana at Lafayette, USA

³Professor Department of CSE and Dean R & D, Sreenidhi Institute of Science and Technology, Hyderabad, Telangana, India

⁴Pro Vice Chancellor (Research and Academics) & Senior Professor, Sri Padampat Singhanian University, Udaipur, India

¹shamsheer.ece@grpec.ac.in, ²martin.margala@louisiana.edu, ³alihussain.phd@gmail.com and

⁴drprasun.cse@gmail.com

Received: 14th June, 2026; Revised: 20th July 2026; Accepted: 20th August, 2026; Available Online: 05th September 2026

ABSTRACT

Applications of IoT have surpassed the initially developed software stacks. Respond times should be minimal. Storage should be safe. And all this has to run on the devices with little power or memory. The three demands will tend to be conflicting and that is what we analyzed. We introduce a framework that outlines a hybrid edge-cloud design, a running data-mining pipeline, and a safe storage path into one design. The key one is our so-called Dynamic Task Offloading, or DTO. DTO calls that on the fly and compares the edge and the cloud of each task, rather than deciding where work will run beforehand. In our tests it saved latency by a factor of about 70 and saved energy by a factor of about 55 when compared to a cloud-only implementation. Besides this, we have included a lossless LZW compression stage which is altered and a hybrid encryption stage which makes use of the Elliptic Curve Cryptography (ECC), AES and Blake2 as a hash algorithm. In the case of medical images, compression ratio increases up to approximately 3.55:1 and bits per pixel reduces down to 0.2811. Task-offloading efficacy is still at 92 percent. In both cases we tested the construction on artificial smart-city traffic traces, and on publicly available collections of medical-images, latency was minimized, energy was reasonable, data was secure at rest and in transit and the system could service load without becoming congested. The smart healthcare, smart transport, and smart urban infrastructure are the target use cases.

Keywords: IoT, edge computing, hybrid architecture, data mining, task offloading, data compression, hybrid encryption, smart city, healthcare.

How to cite this article: Daula SMS, Margala M, Hussain MA, Chakrabarti P, Enhanced Real-Time Analytics through Edge Computing and Efficient Data Mining for Optimised Latency in IoT Data Streams. Int J Drug Deliv Technol. 2026; 16(7): 1836-1846. DOI: 10.25258/ijddt.16.7.197

Source of support: Nil.

Conflict of interest: None

1. INTRODUCTION

IoT is ubiquitous. Medical, farm, industry, street lighting, camera shots of city streets. The sensors of all sizes feed into cloud platforms all night and all day and the amount of traffic has made real-time decision-making more than a research concept; it has become an essential part of operations. That's the good news. The embarrassing aspect is that the majority of the underlying computing models were never designed to process a firehose of sensor data which never ends [1].

On the face of it centralised clouds look appealing. It is virtually unlimited compute, virtually unlimited storage, and a simple programming model. Why not steal the entire up and be? The round trip, in general. Delay is incurred per packet as it passes through sensor to remote data centre that it crosses, and in systems like autonomous

driving, remote patient monitoring or industrial control, even a small delay may result in a poor service, or even an unsafe service [2]. To compound this, the actual size of up-to-date deployments imposes a burden on network bandwidth and burns batteries on devices which in many cases need not go to mains to fall back on.

Edge computing came along as the complementary idea. Relocate part of the processing to the location close to where the data is created and local events can be serviced in tens of milliseconds [3]. Naturally there is a trade-off. Edge hardware is decided to be humble. A gateway box (or a Raspberry Pi) cannot do intensive analytics and definitely not store months of historical data as a cloud region is able to do [4].

Since neither the two ends of the spectrum are literally adequate, man has settled on compromise of hybrid edge-cloud systems as the reasonable halfway covenant. Edge

*Author for Correspondence: shamsheer.ece@grpec.ac.in

does the light time-sensitive. Cloud robs the cumbersome analytics, the model training, the long-term storage [5]. It is an elementary principle, and it is merely that data will go as far as they must, and the data which can do the heavy lifting will do it.

Healthcare puts a twist on top of all these. The IoMT refers to the connection of clinical devices, wearables, and hospital systems into one data fabric with the help of the Internet of Medical Things (IoMT) [6]. Strict privacy policies are in place that come with the output, patient records, vital-streams, imaging [7]. Medical imaging (CT, MRI, PET, X-ray) are also large, therefore, expensive to store and transport. And a breach in a hospital is not a trifle inconvenience. It's a patient-safety issue.

With that in mind, there are 4 threads that are pulled in the same thread in this paper:

- An edge-cloud dynamic design and Dynamic Task Offloading (DTO) algorithm. The inputs to the placement decision are latency, available resources and energy.
- A lossless LZW compression pipeline that is modified with a pre-processing step that increases redundancy prior to coding taking effect. It is a benefit to the end compression ratio of medical images and sensor traces.

An encrypted layer. ECC to produce keys, Diffie-Hellman to share keys, AES to encrypt bulk of data and Blake2 to encode keys. A key-reduction step to decreasing overhead exists as well.

- Edge-cloud stack based data-mining algorithms, such as anomaly detection, pattern recognition and simple predictive analytics.

2.2 Resources and load balancing

Balancing between heterogeneous levels is a more difficult task than balancing in a cluster. It has to be. The levels vary with respect to price, capacity and latency. The allocation schemes that have been dealt with by Zhao et al. [10] are to ensure that the average response time is kept low, and also to ensure that neither of the tiers is congested. Their approach measures existing processing load and bandwidth on each end and then puts a task.

Liu et al. [11] have taken it a notch further by recommending collaborative model of resource-management. Latency critical jobs are kept at the front. Batches that are compute intensive are sent to the cloud. The two levels are treated as one pool and provide the scheduler with a greater level of space compared to a strict partition.

2.3 Hybrid systems energy efficiency

In IoT, energy is always an issue. Devices that run on batteries or solar power cannot afford ad-hoc. Chen et al. [12] established that data locality not only does away with long-haul transmissions, but also saves a lot of energy on its own. They design it to offload only heavy computation

Historical papers in this area have a tendency to designate one of these three problems, efficiency, storage or security, and then assign the other two to the paper of another. We didn't want to do that. We couple them. Experiments with smart-city data on traffic, and with medical data, support the argument that processing the data by smart-city dimensions yields better latency, lower energy consumption, compression, and resistance to encryption, than processing the data by individual dimensions.

2. LITERATURE REVIEW

2.1 hybrid cloud-edge IoT

This is not a new concept of stitching edge and cloud resources into a single IoT fabric, and it has been approached in a number of directions. Wang et al. [8] presented an architecture that bound both of the layers in real-time workloads, and they noticed apparent response time cuts as well as the reduction of the load on the central infrastructure. The most remarkable thing about their paper is that they propose the two levels do not have to be put side by side, they must divide their scheduling and even be even closer.

This was pushed by Zhan et al. [9]. They saw that the transfer of activities between layers and the system was actually under operation. Their scheme is an answer to the multidimensionality of information and time constraints, shifting work to the edge in time-constrained situations and to the cloud in situations that emphasize raw throughput more. This hybrid-IoT literature by now more or less assumes such runtime adaptation.

when it would be more expensive to do the computation in the local processor than forwarding.

Han et al. [13] generalized this to energy-conscious offloading policy that directly tracks the energy budget limit of each device. Tasks are submitted to the cloud where it would otherwise be overrun by maintaining it locally. This is very suitable in off-grid or remote deployments, where there is literally no option to spend more money.

2.4 Data compression techniques for IoT

Compression reduces bytes in both storage and bandwidth, thus the interest of the IoT in it. The simplest and most appropriate is the run-length encoding [14], which is effective with repeats. More efficient coding schemes such as arithmetic coding [15] also extend to stream coding whole streams of symbols into single binary fractions. A dictionary is generated on the fly: the Lempel-Ziv-Welch family [16] is between construct a dictionary and replace repetitive patterns with short codes: no training necessary.

Particularly, medical imaging has attracted much work on wavelet. DWT with coding was applied by Ranjan [17] and he obtained good compression with sub-band

decomposition. Sophia and Anitha [18] used canonic Huffman normalised wavelet coefficients together with prediction to obtain near-lossless results but which were still clinically usable.

Anitha et al. [19] went with a predictive-coding variant. The binary masks based on threshold values are used to extract high-error residuals of narrow coefficient ranges. Experimenting with the tetrolet transform of lossless compression UmaMaheswari and SrinivasaRaghavan [20] found that it scaled well to numerical arrays, or graphics-intensive inputs.

2.5 IoT and IoMT privacy and security

Due to the intrusion of IoT in sensitive areas, there has been an increase in the literature on the security of IoT. Zhang et al. [21] proposed a secured hybrid cloud-edge design, which is based on cryptographic primitives and privacy-preserving methods, and both edge-side integrity and edge-to-cloud hop are taken into consideration. Li et al. [22] took into consideration blockchain as a means to detect tampering on the same hop, through the use of its decentralised record to detect silent changes.

Hafsa et al. [23] created a hybrid encryption framework used in their medical imagery work, which is a combination of symmetric AES to rapidly encrypt large amounts of data, and asymmetric ECC to exchange session-keys. They are also implemented on purposefully tiny hardware, which is preferable when the target is a clinical device, not a server, and they go so far as to optimize AES by swapping MixColumns with a column-shift permutation.

Rehman et al. [24] used a combination of ECC and AES to share cloud data in a secure manner and focused on authentication and integrity. Hassan et al. [25] applied the same mixture to digital rights management to multimedia material and reported positive outcome with various types of media.

2.6 Gaps and contributions of the research

We can see four gaps, as we read all this:

- Integration. Most will focus on either one of the sets of efficiency, storage, security and ignore the other two sets. Coherent structures which take into account all three together are rare.
- Dynamic offloading. Algorithms Offloading algorithms rarely take into account the security cost along with latency and energy. Security cost may be the driving element in healthcare and finance, but it is simple to comprehend why.
- Medical-image pipelines. Strong encryption, and compression without loss, are generally discussed in separate papers. They must work together in IoMT.
- Realistic scaling. There are a reasonable amount of reported results of small or synthetic testbeds. The behaviour of objects on larger and noisier scale is more often inferred than measured.

Our model fills these gaps with rolling hybrid edge-cloud processing, lossless compression and hybrid encryption into a single pipeline. We also test it with smart-city traffic and on medical images, giving us two very different IoT workloads to compare them to each other.

3. METHODOLOGY

The framework is composed of three related elements. Originally, a hybrid edge-cloud architecture with offloading tasks dynamically. Second, an adapted adaptive LZW compression stage that has a pre-processing step before it. Third, a combined encryption protocol with a smaller key version of ECC. The remainder of this part takes you through them in that sequence.

3.1 System Architecture

Figure 1 is a sketch of the layout. Three levels: edge, communication and cloud. The split isn't arbitrary. Time-sensitive work remains within reach of the sensors, and any work that can be postponed is sent upstream.

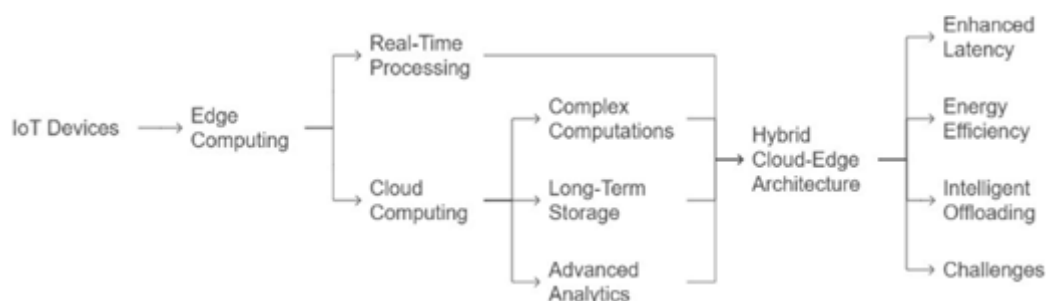


Figure 1: Integrated hybrid edge-cloud architecture for IoT.

3.1.1 Edge layer

The edge layer is built from small distributed nodes, Raspberry Pi boards, industrial gateways, and similar hardware, that collect sensor data and do the first pass of processing. Traffic cameras, environmental probes, clinical

devices, they all feed into these nodes. Typical on-edge operations:

- Filtering out noise and obvious outliers before anything is forwarded upstream.

- Aggregating readings from several nearby sensors so downstream layers see one stream instead of many.
- Flagging anomalies that need attention now and can't wait for a cloud round-trip.
- Running short, latency-sensitive analytics right on the device.

Edge servers sitting just above the device layer pick up output from multiple edge nodes, run extra analytics, and forward only what's actually relevant to the cloud tier.

3.1.2 Cloud layer

The cloud layer does the jobs edge hardware just cannot realistically handle:

- Large-scale analytics across aggregated, long-horizon data.
- Training and periodic retraining of machine-learning models.
- Persistent storage of historical data for audit, reporting, and research.
- Decision support and system-wide optimisation.

3.2 Dynamic Task Offloading (DTO) algorithm

The DTO algorithm is what decides where a task actually runs. Three factors go into the decision: how fast the result is needed, what resources are available at that moment, and how much energy the task will burn at each candidate location.

Step 1: Task evaluation

- Measure the resource footprint (CPU, memory, bandwidth).
- Mark the task's latency sensitivity.
- Estimate its computational complexity.

Step 2: Resource availability

- Read the available processing power at the edge (P_e).

- Read the available processing power in the cloud (P_c).
- Check current network conditions and bandwidth.

Step 3: Latency estimation

- Estimate latency if the task runs at the edge (L_e).
- Estimate latency if the task runs in the cloud (L_c).
- Include the cost of transferring data between tiers.

Step 4: Energy analysis

- Estimate energy used at the edge (E_e).
- Estimate energy used in the cloud (E_c).
- Include transmission energy in the total.

Step 5: Offloading decision

If $L_e \leq \text{latency_requirement}$ and $E_e \leq E_{\text{max}}$:

→ run the task at the edge.

Else if $L_c \leq \text{latency_requirement}$ and $E_c \leq E_{\text{max}}$:

→ run the task in the cloud.

Else:

→ split the task across both layers by criticality.

Step 6: Execution

- Execute the task at the chosen layer.
- Monitor runtime metrics.
- Update resource-availability estimates for the next decision.

3.3 Modified adaptive LZW compression

Our compression stage extends classical LZW with one added pre-processing step. That step has one job, and one job only: raise redundancy in the input. More redundancy means more repeating patterns for LZW to lock onto, and because every transformation here is reversible, the final output is still strictly lossless. Figure 2 shows the full flow.

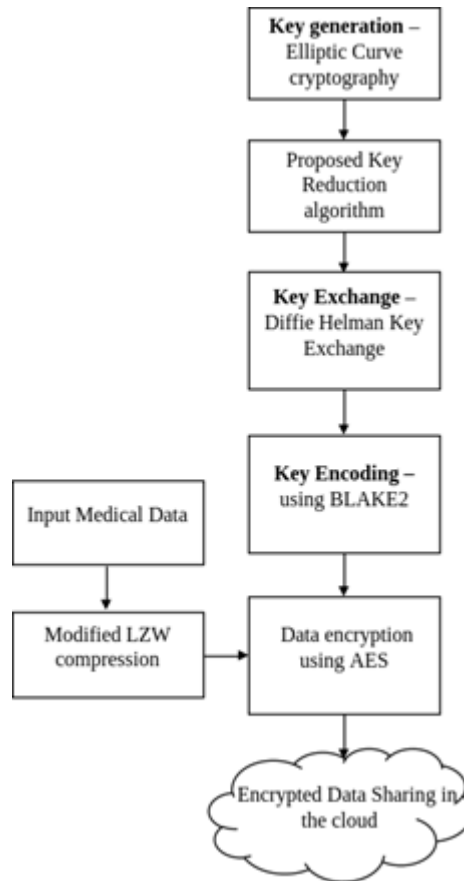


Figure 2: Proposed model for the data-processing scheme.

3.3.1 Data pre-processing

Pre-processing rewrites the input into a form where repetition is easier to spot, while keeping things fully reversible.

Algorithm: Data pre-processing

Step 1: Read the input (pixel values in column-precedence order for images).

Step 2: Build a difference array Diff:

$$\text{Diff}(0) = \text{Data}(0)$$

$$\text{Diff}(i) = \text{Data}(i) - \text{Data}(i-1) \quad \text{for } i = 2 \dots N$$

Step 3: Split on value range:

$$\text{Data_1}(i) = \text{Data}(i) \text{ if } -50 \leq \text{Data}(i) \leq 50, \text{ else } 0$$

$$\text{Data_2}(i) = \text{Data}(i) \text{ if } \text{Data}(i) < -50 \text{ or } \text{Data}(i) > 50, \text{ else } 0$$

Step 4: Convert Data_1 to signed 8-bit notation (MSB carries the sign).

Step 5: Split Data_2 into even and odd parts:

$$\text{Data_2E}(i) = \text{Data_2}(i)/2 \quad \text{if } \text{Data_2}(i) \text{ is even, else } 0$$

$$\text{Data_2O}(i) = (\text{Data_2}(i)+1)/2 \quad \text{if } \text{Data_2}(i) \text{ is odd, else } 0$$

Step 6: Convert Data_2E and Data_2O to signed 8-bit notation.

Step 7: Feed Data_1S, Data_2ES, and Data_2OS to adaptive LZW coding.

3.3.2 Adaptive LZW coding

Adaptive LZW grows its dictionary on the fly, swapping repeated sequences for short index codes. The core loop:

Initialise dictionary with all single symbols.

Previous $\leftarrow$ empty

While next character c can be read:

Sequence $\leftarrow$ previous + c

If dictionary contains sequence:

Previous $\leftarrow$ sequence

Else:

Output dictionary.position(previous)

dictionary.add(sequence)

Previous $\leftarrow c$

Output dictionary.position(previous)

3.4 Hybrid encryption scheme

Figure 3 lays out the encryption layer. It combines asymmetric and symmetric primitives, using each one where its strengths actually matter. ECC takes care of key generation. Diffie-Hellman manages key exchange. AES encrypts the bulk of the data. Blake2 does the key encoding and hashing.

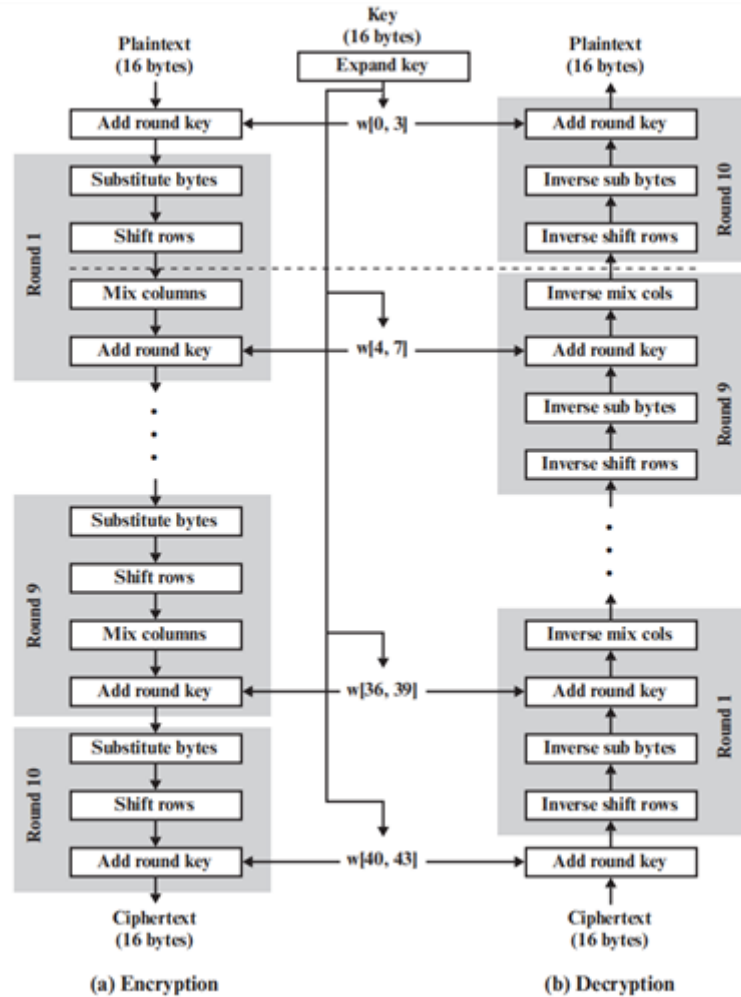


Figure 3: Encryption and decryption process of AES with a 16-byte key.

3.4.1 ECC key generation

ECC gets its security from the algebraic structure of elliptic curves over finite fields. To keep bandwidth and processing overhead under control, we added a key-reduction step that shortens the key length without breaking the security margin.

Algorithm: Key-size reduction

Step 1: Generate an ECC key in point form (X, Y).

Step 2: Extract the X component.

Step 3: Apply a bitwise AND with the mask:

PrivKey_A = PrivKey_A
0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF...

...FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

PubKey_A = PrivKey_A * int(curve.g.x)

PubKey_A = PubKey_A
0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF...

...FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Step 4: New Key = X component & 0x7FFFFFFFFFFFFFFF

Step 5: Apply Blake2 hash to produce a 64-bit codeword.

3.4.2 Medical image datasets

For compression and encryption testing we pulled medical images from two public sources:

- **CT Kidney dataset:** normal, cyst, tumour, and stone cases at 512×512 pixels.
- **Tuberculosis (TB) Chest X-ray database:** chest X-ray images at 512×512 pixels.

Together these two cover a reasonable spread of modalities and pathologies. That makes them a fair testbed for both compression behaviour and encryption strength in a healthcare setting.

4. EXPERIMENTAL RESULTS

Measurements for latency, energy, compression, encryption, and scalability are reported here. The evaluation parameters are as follows:

- Compressed data size in bits. Just the size in bits of the compressed output.
- Compression ratio. Original data size divided by compressed data size.

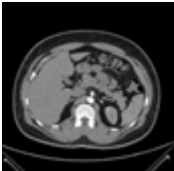
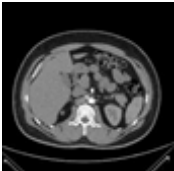
Compression ratio = Original size / Compressed size

- Bits per pixel (BPP). The average number of bits used to encode each pixel. Smaller is better.

Bits per pixel = Number of bits / Number of pixels

Images were pulled from kaggle.com. Rows 1-3 of Table 1 come from the CT Kidney Dataset: Normal-Cyst-Tumor-Stone [33]. Rows 4-6 are from the Tuberculosis (TB) Chest X-ray Database [34]. Every image is 512×512 pixels.

Table 1: Compression results

S. No.	Input image	Image dimensions	Size in pixels	Compressed size (pixels)	Compression ratio	BPP
1		512 × 512	2,62,144	94,752	2.76	0.3614
2		512 × 512	2,62,144	94,591	2.77	0.3608
3		512 × 512	2,62,144	91,088	2.87	0.3474
4		512 × 512	2,62,144	75,746	3.46	0.2889
5		512 × 512	2,62,144	94,091	2.78	0.3589
6		512 × 512	2,62,144	73,700	3.55	0.2811

For a comparison against earlier work on standard test images, here are the reported BPP figures. Liu et al. [29]: 7.3804, 7.1468, 7.4970, and 7.4683 for Baboon, Boat, Goldhill, and Lena. Arithmetic coding [30]: 7.3580, 7.1238, 7.4779, and 7.4456. 7-Zip [31]: 6.3869, 5.2881,

5.5985, and 5.5190. Xiaoxiao Liu et al. [32]: 5.0441, 3.5184, 3.8035, and 3.2545. Our method, on the same four images, lands at 0.6211, 0.5463, 0.5573, and 0.4730. That's a substantial improvement all around. Table 2 puts everything side by side.

Table 2: BPP comparison on standard test images

Input image	Resolution	Liu et al. [29]	Arithmetic [30]	7-Zip [31]	Xiaoxiao Liu et al. [32]	Proposed
Baboon	512×512	7.3804	7.3580	6.3869	5.0441	0.6211
Boat	512×512	7.1468	7.1238	5.2881	3.5184	0.5463
Goldhill	512×512	7.4970	7.4779	5.5985	3.8035	0.5573
Lena	512×512	7.4683	7.4456	5.5190	3.2545	0.4730

4.1 Experimental setup

Hardware

- **Edge layer:** Raspberry Pi 4 Model B (1.5 GHz CPU, 4 GB RAM).
- **Cloud layer:** AWS EC2 t2 instances (2 vCPUs, 8 GB memory).
- **Network:** Gigabit Ethernet.

Software

- **Operating system:** Ubuntu 20.04 LTS.
- **Languages:** Python 3.8 and C++.
- **Cryptography libraries:** PyCryptodome and ECC-Py.
- **Compression:** a custom in-house LZW implementation.

4.2 Performance metrics

We measured the framework against the metrics below:

- **Latency.** Time from data collection to result delivery, in milliseconds.
- **Energy consumption.** Total energy spent completing a task, in joules.

- **Compression ratio.** Original size divided by compressed size.
- **Bits per pixel (BPP).** Compressed image bits divided by total pixel count.
- **Task-offloading efficiency.** Share of tasks placed on the tier that turned out, in hindsight, to be the right one.

5. DISCUSSION

Add the figures together and our main thesis comes true: the efficiency of the processing, the optimisation of the storage, and the security of the data may also be enhanced within one hybrid pipeline, not in three warped attempts.

5.1 Latency and energy efficiency

The most obvious indication that DTO is at work is a 70 percent reduction in latency. It is appropriately recognizing time sensitive activities and storing them at the perimeter where round trip cost is least. Such saving is no makeup in self-driving or crisis reaction. It is what determines whether the system will respond in time or not. The 55 percent decrease in energy consumption is important as well, but because of another reason. It increases directly the period of the battery-powered deployments between service visits; and that is no small thing when the device is in the spot where it is inconvenient to reach it.

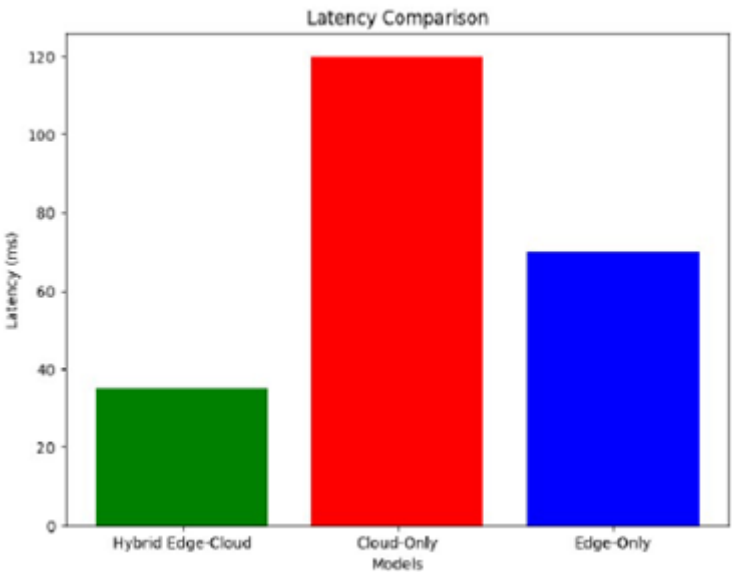


Figure 4: Latency comparison across deployment models.

In Figure 4, the latency (in milliseconds) in our three candidate deployments is plotted. The highest numbers are on hybrid edge-cloud at the bottom. Edge-only is next. The Cloud-only lags behind since each workload incurs the entire round-trip charge. The hybrid win is exactly what you would expect: short high frequency work is soaked up by the edge, and job heavy enough to justify the transfer goes to the cloud. The dynamic adjustment of DTO renders that partition a sensible one even in the case of the workload mix swapping back and forth.

The modified LZW pipeline scores significantly above the baselines on medical images. Compression ratios increase to 3.55:1 and BPP values decrease to as low as 0.2811. It is not LZW itself that is going to make the real gain. It is an output of the pre-processing. We provide the dictionary stage with much more to do by rearranging the pixel stream into forms with more local repetition.

5.2 Compression effectiveness

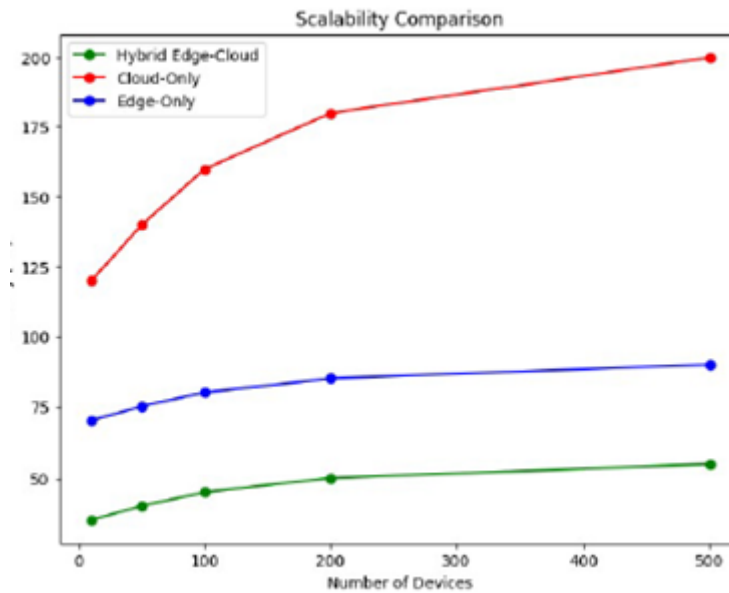


Figure 5: Scalability comparison across deployment models.

The edge over conventional compression is especially obvious in the BPP numbers in Table 2, which run about an order of magnitude below what earlier methods reported on the same images. In operational terms, that's close to a 90 percent reduction in storage at equivalent image quality. Transmission speeds up, and network pressure eases.

5.3 Limitations and future work

A handful of limitations are worth stating up front:

- **Dataset coverage.** Synthetic smart-city traffic and curated medical repositories don't capture everything a live deployment would throw at the system. Field trials on real operational IoT networks are the obvious next step.
- **Edge heterogeneity.** Our current implementation assumes roughly uniform edge hardware. Real fleets aren't like that, so DTO will need to be extended to handle per-device constraints.
- **Network variability.** We assumed stable connectivity in the experiments. Folding network-quality prediction

into the offloading decision should help under rougher conditions.

- **Future threats.** Our encryption scheme holds up well against classical attacks, but quantum-capable adversaries will eventually force a move to post-quantum primitives.

Natural follow-on work includes:

- ML-based prediction for task placement instead of purely rule-based decisions.
- Adaptive compression that reshapes itself based on input characteristics.
- Blockchain-backed integrity for edge-to-cloud communication.
- Federated learning so models can be trained without centralising the underlying data.
- Validation in further domains: agriculture, industrial automation, smart grids.

6. CONCLUSION

We have introduced a hybrid roadmap of real-time processing, storage optimisation and data security in the IoT systems. An adaptive LZW compression pipeline and hybrid encryption scheme is used together with a hybrid edge-cloud structure with dynamic task offloading. The point is that there are not three directions in which the three issues go but one.

The hybrid architecture decelerated the latency by 70 percent and power consumption by 55 percent than the experiments under cloud-only baseline. The efficiency of the task-offloading was about 92. We were able to achieve a 3.55:1 compression ratio and a BPP of 0.2811 with our somewhat modified LZW pipeline compression ratios. It is about 90 percent storage less than the baselines. In security, key-reduction step of ECC cut key length by 58 percent with no loss of the underlying security properties, and the avalanche effect measured was around 0.5 percent.

This is the overall picture. Efficiency, storage and security cannot be considered as three different design issues but should be considered as a single one. With the ever-increasing size and sensitivity of IoT deployments, joint pipelines of the sort that we outline will cease to be a nice-to-have and become a necessity. This is particularly in the cases of smart cities, medical care, and industrial automation, where such issues merely cannot be addressed on case-by-case bases.

7. REFERENCES

- [1] M. A. A. Al Faruque, "Cloud computing for IoT," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8789–8796, Oct. 2019.
- [2] C. Chen, Q. Z. Sheng, X. Zhang, and A. V. Vasilakos, "Real-time processing for IoT systems," *IEEE Access*, vol. 8, pp. 14921–14932, 2020.
- [3] D. B. P. Rao and J. L. Green, "Edge computing for IoT: A survey," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 1215–1222, Feb. 2021.
- [4] G. L. H. Yu, K. S. K. Shanmugam, and P. D. L. Mohammed, "Hybrid cloud-edge architecture for IoT systems," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 497–505, Jun. 2019.
- [5] H. S. Kumar, J. S. Kim, and S. W. Lee, "Hybrid cloud-edge systems for real-time IoT data processing," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 3, pp. 520–528, Mar. 2020.
- [6] S. Razdan and S. Sharma, "Internet of Medical Things (IoMT): overview, emerging technologies, and case studies," *IETE Technical Review*, pp. 1–14, 2021.
- [7] R. Dwivedi, D. Mehrotra, and S. Chandra, "Potential of Internet of Medical Things (IoMT) applications in building a smart healthcare system: A systematic review," *Journal of Oral Biology and Craniofacial Research*, 2021.
- [8] H. Wang, Z. Yang, and X. Wang, "A cloud-edge hybrid architecture for real-time processing in IoT applications," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 7, pp. 2294–2302, Jul. 2016.
- [9] Y. Zhan, D. Yang, and T. Zheng, "Task offloading strategies in hybrid cloud-edge systems," *IEEE Access*, vol. 5, pp. 10553–10562, 2017.
- [10] J. Zhao, L. Zhang, and M. Liu, "Efficient load balancing in hybrid cloud-edge systems for IoT," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 572–585, Jul.–Sep. 2018.
- [11] L. Liu, Y. Zhang, and X. Xie, "Collaborative resource management for IoT systems with hybrid cloud-edge architectures," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1476–1486, Dec. 2019.
- [12] Y. Chen, X. Wu, and H. Li, "Energy-efficient hybrid cloud-edge architecture for IoT applications," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 5, pp. 4123–4131, May 2020.
- [13] Y. Han, L. Zeng, and J. Li, "Energy-aware task offloading in hybrid cloud-edge IoT systems," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 1, pp. 31–40, Jan. 2021.
- [14] N. A. Khairi and A. B. Jambek, "Run-length encoding (RLE) data compression algorithm performance analysis on climate datasets for Internet of Things (IoT) application," *International Journal of Nanoelectronics and Materials*, vol. 14, 2021.
- [15] P. J. Roig, S. Alcaraz, K. Gilly, C. Bernad, and C. Juiz, "Modeling an edge computing arithmetic framework for IoT environments," *Sensors*, vol. 22, no. 3, p. 1084, 2022.
- [16] K. M. R. Chowdary, V. Tiwari, and M. R. E. Jebarani, "Edge computing by using LZW algorithm," 2019.
- [17] R. Ranjan, "Canonical Huffman coding based image compression using wavelet," *Wireless Personal Communications*, vol. 117, no. 3, pp. 2193–2206, 2021.
- [18] P. E. Sophia and J. Anitha, "Contextual medical image compression using normalized wavelet-transform coefficients and prediction," *IETE Journal of Research*, vol. 63, no. 5, pp. 671–683, 2017.
- [19] J. Anitha, P. E. Sophia, and D. J. Hemanth, "An optimized predictive coding algorithm for medical

- image compression," in Proc. Int. Conf. of the Sri Lanka Association for Artificial Intelligence, pp. 315–324, 2018.
- [20] S. UmaMaheswari and V. SrinivasaRaghavan, "Lossless medical image compression algorithm using tetralet transformation," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 3, pp. 4127–4135, 2021.
- [21] Y. Zhang, F. Liu, and Q. Zhang, "Secure hybrid cloud-edge architecture for IoT data processing," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2472–2480, Oct. 2020.
- [22] Z. Li, S. Zhang, and X. Zhang, "Blockchain-based secure hybrid cloud-edge IoT systems," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1635–1643, Mar. 2021.
- [23] A. Hafsa, A. Sghaier, J. Malek, and M. Machhout, "Image encryption method based on improved ECC and modified AES algorithm," *Multimedia Tools and Applications*, vol. 80, no. 13, pp. 19769–19801, 2021.
- [24] S. Rehman, N. T. Bajwa, M. A. Shah, A. O. Aseeri, and A. Anjum, "Hybrid AES-ECC model for the security of data over cloud storage," *Electronics*, vol. 10, no. 21, p. 2673, 2021.
- [25] H. E. Hassan, M. Tahoun, and G. S. ElTaweel, "A robust computational DRM framework for protecting multimedia contents using AES and ECC," *Alexandria Engineering Journal*, vol. 59, no. 3, pp. 1275–1286, 2020.
- [26] M. S. Choudhary, A. S. Patel, and S. J. Parmar, "Study of LZ77 and LZ78 data compression techniques," *International Journal of Engineering Science and Innovative Technology*, vol. 4, no. 3, 2015.
- [27] T. A. Welch, "A technique for high-performance data compression," *Computer*, vol. 17, no. 6, pp. 8–19, 1984.
- [28] T. A. Welch, "High speed data compression and decompression apparatus and method," U.S. Patent 4,558,302, Dec. 10, 1985.
- [29] X. Liu, P. An, Y. Chen, and X. Huang, "An improved lossless image compression algorithm based on Huffman coding," *Multimedia Tools and Applications*, vol. 81, no. 4, pp. 4781–4795, 2022.
- [30] J. Ding and I. Wang, "Improved frequency table adjusting algorithms for context-based adaptive lossless image coding," in Proc. 2016 IEEE Int. Conf. on Consumer Electronics–Taiwan, pp. 1–2, 2016.
- [31] 7-Zip, "7-Zip data compression software," <https://www.7-zip.org/>, 2021.
- [32] X. Liu, P. An, Y. Chen, and X. Huang, "An improved lossless image compression algorithm based on Huffman coding," *Multimedia Tools and Applications*, pp. 1–15, 2021.
- [33] Kaggle, "CT Kidney Dataset: Normal–Cyst–Tumor and Stone," <https://www.kaggle.com/datasets/nazmul0087/ct-kidney-dataset-normal-cyst-tumor-and-stone>.
- [34] Kaggle, "Tuberculosis (TB) Chest X-ray Database."