ECG Task Performance Tuning and IoT Data Compression Optimization Through Dynamic-Deep Learning

K Divya Lakshmi[1,] P Subbaraidu [2] G V Sanjeeva Reddy[3], R Aruna [4]

[1,2,3,4] Asst , Professor, Department of ECE, K. S. R. M College of Engineering(A), Kadapa

Abstract—

One common use of IoT gadgets is the monitoring of medical data like Electrocardiogram (ECG) signals. To save storage and transfer costs, the vast amounts of sensor recordings are often compressed before being sent to the Cloud. Although a lossy compression results in a high compression gain (CG), this approach may hinder the efficiency of an ECG utility (a downstream task) due to information loss. Previous ECG tracking efforts have focused either on improving the task's overall performance or on enhancing the sign reconstruction. Instead, we provide a self-adaptive lossy compressSion solution that lets you choose your preferred overall performance stage for your downstream commitments while still keeping your CG optimized to lower your Cloud service costs. For IoT-Cloud systems, we suggest using Dynamic-Deep, a compression method that takes into account individual tasks. Out of a wide range of downstream duties, our compressor is able to optimize the CG while still meeting the performance requirements. In deployment, the IoT component tool adjusts the compression and transmits an optimum illustration for each data segment, taking into consideration the preferred overall performance of the downstream activity without needing to poll the Cloud for feedback. We conduct a thorough evaluation of our technique using well-known ECG programs and ECG datasets that include the coronary heart rate (HR) arrhythmia type.

INTRODUCTION

Internet of Things (IoT) devices are widely used to monitor and send sensor data to the Cloud for centralized storage and execution of downstream tasks. For example, hospitals use IoT medical devices to constantly monitor Electrocardiogram (ECG) signals, the patient heart's activity over time. Several downstream tasks make use of ECG signals: For alerting the staff of HR arrhythmias [19] or extracting indicative features (e.g., R-R peaks [22]) for the purpose of heart disease diagnostics

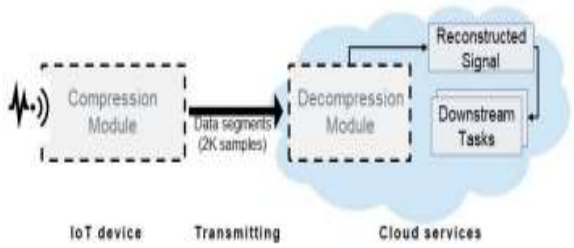


Fig. 1: Typical architecture of a modern IoT-based medical monitoring system with data compression.

TABLE I: Evaluate HR arrhythmia classification and reconstriction tasks on 602 test segments after compression.

Compression Method	CG	Avg. signal reconstruction error [%]	HR classification F1-score (Precision,Recall)	Violation of upper bound** [%]
LZ77 (Lossless)	2.7	0	**0.87*** (0.87,0.87)	**0**
CAE (SOTA)	32.25	**2.73**	0.20 (0.32,0.15)	10.21
Dynamic-Deep #1	48.31	3.81	0.73 (0.72,0.74)	**0**
Dynamic-Deep #2	**52.2**	4.2	0.69 (0.698,0.697)	**0**

* The HR arrhythmia classification F1-score (Precision,Recall) results when applied to uncompressed data segments

** Percentage of data segments that experience classification error above 0.75

Such medical monitoring settings generate large amounts of continuous sensor data to be sent to the Cloud. Transmitting the raw signal would imply power-hungry devices and high processing and storage Cloud costs. Therefore, an effective data compression scheme is required to reduce the transmission and storage requirements. An efficient compression module is typically deployed on the device to accommodate settings with low power resource-limited IoT devices, while a decompression module is deployed in the Cloud. Compression gain (CG) is usually used to evaluate such data compression schemes by calculating the division sizes of the original representation against the compressed representation. Fig. 1 presents an end-to-end data flow in a Cloud-based monitoring system: compressed data segments are prepared for transmission, and the data is decompressed back to raw ECG signal in the Cloud for further processing and analysis.

RELATED WORK

Existing compression strategies had been tailored into clinical IoT environments, generally to in shape the low energy requirements [21], [8]. Such techniques are break up into lossless [23] and lossy [8] categories. While lossless compression is ideal for preserving the integrity of signals like ECG [16], lossy compression is fairly environmentally friendly (in reducing storage needs), making it suitable for IoT sensing. Compression using a transform transform. Transformation-based compression, which aims to preserve the signal's important elements inside the transformed region, is a commonplace conventional approach to lossy compression. Few exquisite examples are Fourier rework [20], Wavelet rework [3] or the Cosine rework [1], every with its very own area transformation preference. The trans fashioned illustration of ECG records is frequently sparse, and therefore retaining the proper components of the illustration allows one to get a reconstruction sign of probably excessive fidelity [8], [4]. In [21], it is advised against using a dynamic scheme that takes the edge at the length of the example given. The essential disadvantage of the rework-primarily based totally method is the usage of a predefined area transformation, which won't cause the best CG for the favoured reconstruction level. Neural network (NN) primarily based totally compression. Finding the best area transformation for the compressed illustration is a tedious process, but NNs make it automatic. Auto-Encoders (AE), an own circle of relatives of NNs, had been appreciably studied [14], [12] and proven to efficiently research an expressive-yet-green illustration of ECG segments, and therefore offer a better CG than rework primarily based totally compression techniques [21].

To achieve SOTA compressSion results, recent work [24] employs a convolutional AE with 27 layers. As a rule, AE designs are limited to the tractable CGs due to their single constant compression level. Variable-charge compression. Recent works cautioned NN architectures with a couple of compression ranges permit balancing among CG and reconstruction quality [6] [2]. However, they require to outline earlier a charge manipulate parameter. Aditonally, they consciousness their assessment on reconstruction quality, therefore there may be no assure on downstream responsibilities overall performance. Whereas Dynamic-Deep is tuned to a favoured downstream responsibility` overall performance and optimizes CG for every records section robotically in place of guide parameter changes.

MOTIVATION FOR BALANCING CG AND DOWNSTREAM TASKS PERFORMANCE

Lossy compression methods, such as SOTA CAE [24], are based on an architecture with a single fixed compression level. However, a fixed compression level may not necessarily satisfy a desired bound on the task's error for every data segment. Fig. 3 presents the CCE quartiles across ECG tested segmints for different compression levels: 64, 32, 16 and none. (See section VI-A for details on the task and dataset used). Denote Coax as an extended SOTA CAE implementation with a CG of xx, and let CAE0 represent no compression. We note that non zero losses may occur in uncompressed operations due to the inherent error of the HR detection model. Let an example scenario be when the admin bound the HR arrhythmia classification loss (CCE) to 0.75, none of the fixed compressors can satisfy this bound for all segments, as shown in Fig. 3. To meet the bound of 0.75 we can apply an approach of 2-compression levels by combining a single compressor and no compression. For instance, with a single compressor CAE32, 75% of data segments meet the upper bound, hence the rest 25% remain non compressed. Such a setting yields an average CG of 24. Additionally, we can leverage higher CG as long as those meet the upper bound error of desired downstream task. For instance, CAE64 meets the upper bound for approximately more than 70% of data segments. Therefore, increasing to 3-compression levels allows more compression possibilities with potential of higher CG. Back to the example meeting the bound of 0.75, approximately 75% of data segments can be compressed with a high CG of 64 or 32 while the rest 25% remain uncompressed, reaching higher CG of 48.31.

DYNAMIC-DEEP HIGH-LEVEL DESIGN

To receive feedback from the downstream tasks, we propose a solution that extends the classical CAE architecture with multiple compression levels in a task-aware fashion. DynamicDeep consists of the following (as shown in Fig. 2):

Compression Module:

set of encoders that compress every raw data segment into multiple compression level representations. To reduce the need for actual feedback from the downstream tasks, we employ a dense layer trained to predict the downstream tasks' weighted error for each compression level. We choose the highest

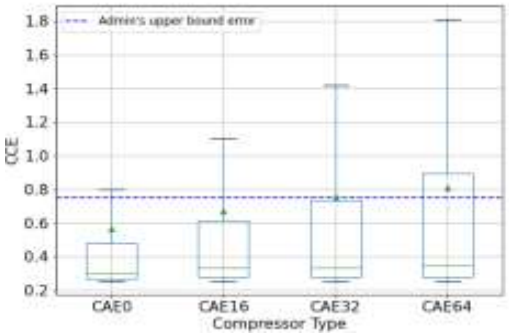


Fig. 2: Categorical cross entropy loss quartiles for HR arrhythmiaMia classification over test ECG data segments (of size 602) feasible CG based on the pre-configured upper bound error and the feedback prediction.

Decompression Module:

set of decoders that accept multiple representations of different compression levels and reconstruct the data segment. For optimizing the joint performance of downstream tasks and reconstructed tasks, the training phase requires differenttable downstream tasks (e.g., NN-based). Nonetheless, after the training phase, the reconstructed signal (from the pretrained compressor) allows executing additional downstream tasks not necessarily differentiable. We found a low correlation between reconstruction error and downstream tasks' error (see Section V-D). We therefore designed Dynamic-Deep to predict the downstream tasks' error rather than the reconstruction error as a proxy for the error feedback.

DYNAMIC-DEEP IMPLEMENTATION DETAILS

A. Multiple Compression Levels The following architectural changes were made to extend the CAE32 to support 3-compression levels (64, 32 or no compression). We chose 3-compression levels based on experiments comparing the performance (and footprint) of different numbers of compression levels, including CAE16. We obtained the highest impact with 3-compression levels (Full implementation details in [10]). Denote conv(a,b) as a convolutional layer with x number of filters, kernel size of y, and Up sample(z) as an up-sample layer with size z. To support CG of 64, convolutional layers conv(64,7) and conv(1,3) were added after CAE32 encoder's layer number 12, yielding an output shape of (31,1). A decoder adapter transforms the compressed representton of 64 to input the CAE32's decoder. The adapter has four layers: conv(16,7), conv(32,3), Up sample(2), conv(32,3), Up sample(2) with an output size of (124,32). The output of the adapter is the input of layer 18 in CAE32. Each compressed representation has its dense layer supporting the downstream tasks' prediction. The dense layer's output shape is (1,1) with a REL activation.

Minimizing Memory Footprint for IoT Support

Deploying compressing modules on lightweight IoT devices requires a resource-constrained implementation. We use twoencoders to support 3-compression levels, which results in a 1.06MB memory footprint for the extended SOTA CAE. We use several techniques to reduce the model's memory size:

Deep learning compression techniques:

The number of learnable parameters of the CAE's 10th layer is reduced by decreasing the number of filters and kernel size. We compensate and preserve compression performance by preserving the receptive field using convolutional layers with stride=2 instead of pooling layers [5]. Note that the encoder's 10th layer is used in all compression levels.

Sharing layers:

Each encoder (CAE32, CAE64) shares the same learnable parameters;therefore, memory and computation are reused to avoid linear memory increase for each compression level [13]. Only the last encoder's layers of each compression level are computed uniquely. Table II summarizes the resulting memory size when applying the above techniques. It decreases the number of parameters to only 84K parameters which are 67% fewer parameters to the straightforward 3-compression level CAE's encoder.

Combined Loss Function

Dynamic-Deep has three loss functions accumulated to a combined loss function. The reconstruction loss LR, calculates the distance between each sample in the original data segment X and the reconstructed data segment Xˆ over M samples.

$$L_R = \frac{1}{M}\sum_{i=0}^{M-1}\frac{|X[i]-\hat{X}[i]|}{X[i]} * 100 \qquad (1)$$

The downstream task weighted error Law, accumulates the downstream tasks' loss functions. Let it denote task I, and Lit and wi denote its loss function and the weighted (scaling) factor of the loss function, respectively.

$$L_w = \sum_i w_i * L_{t_i} \qquad (2)$$

The combined loss function Lc, combines the reconstruction loss LR with the downstream tasks' weighted error Lw. w0 scales the LR to balance between reconstruction performance and downstream tasks performances.

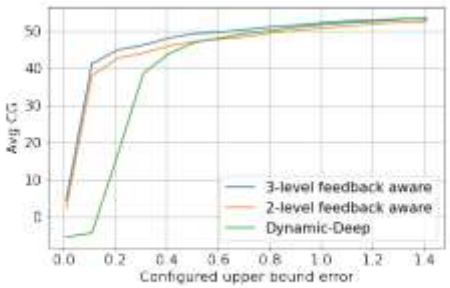
$$L_c = w_0 * L_R + L_w \qquad (3)$$

Finally, Dynamic-Deep learns to predict Lw using the mean squared error (MSE).

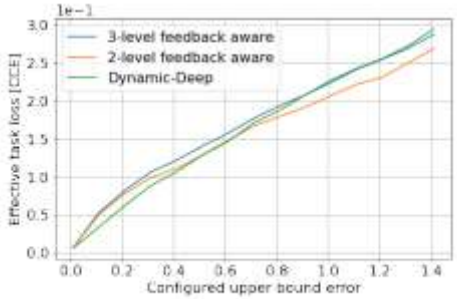
Training

We observe a low correlation between the tested downstream tasks and the reconstruction errors across a wide range of the weighting factors wi (see tech report [10]). TABLE II: Compression Module's Memory minimization

	Number of parameters	memory size [KB]
Original 2-compression level (CAE based)	133K	532
Original 3-compression level (CAE based)	266K	1064
Deep learning compression techniques	168K	678
Sharing layers (**Dynamic-Deep 3-compression level**)	**83K**	**332**



(a) Average CG against pre-configurable bound



(b) Effective loss against pre-configurable bound

Hence, training a downstream task in isolation on the reconstructed signal may result in limited performance. We thus train Dynamic-Deep in three phases, where the second phase is repeated for each additional downstream task. First, we train the compressor to optimize the reconstruction task (see eq. (1)). Here, we use the MIT-BIH dataset by applying the preprocessing described in [24]. Second, we fine-tune with cascaded downstream tasks loss, including reconstruction loss (see eq. (3)). The downstream tasks' models' weights are frozen and trained for additional 20 epochs with the CinC dataset by applying the preprocessing described in [19]. Finally, training the dense layer to predict the downstream tasks' weighted error for additional 10 epochs. As before, the compressor is frozen to not penalize previous training phases.

EXPERIMENTAL RESULTS

Datasets and Downstream Tasks Datasets:

used for training and evaluation:

• MIT-BIH [17]: used to evaluate ECG compression as it includes different types of noise patterns and various shapes of arrhythmic QRS complexes [15]. The benchmark contains 48 half-hour ambulatory ECG recordings yielding a data set comprising 4.8K ECG data segments.

• CinC [7]: Captured from the AliveCor ECG monitor and contains about 7K records with 8960 samples each. These records are annotated by medical experts to the following classes: Atrial Fibrillation(AF), noise, other rhythms or normal. The test set has 602 data records where 40% are abnormal events. Training set has 5.4K records. Downstream tasks: (NN based) architectures were chosen:

• HR arrhythmias classification: implemented using convolutional NN (CNN) [19] as a classification task. The ground truth (GT) of this task uses labels from the CinC dataset. Those labels are annotated by medical experts.

• R-R peak extraction(RPnet): implemented using NN [22] as a regression task locating the position in time of the peak. The GT of this task (i.e. R-R peaks on raw ECG) is generated by running the RPnet NN on the raw ECG signals offered by the CinC dataset. Then each compression level is evaluated relatively to the GT. Both are commonly used tasks in real-world ECG applications.

Task Awareness Evaluation

We focus our evaluation on the Dynamic-Deep downstream tasks' predictions performed by the dense layer in the compression module. We compare our predictions against two theoretical models, in which the downstream task feedback is available for the compressor:

1) 2-level feedback-aware: the method has 2-compression levels of 64 or 1 (act as a lower bound).

3-level feedback-aware: the method has 3-compression levels of 64, 32 or 1 (act as an upper bound).

Each model executes the downstream tasks for every data segment and measures the error at each compression level. Then, they choose the highest compression that meets the configured upper bound error. If none meets the upper bound, the no compression level is chosen. Note that such a method is not applicable in typical IoT settings since the feedback is not readily available at the edges. We evaluated Dynamic-Deep vs. the theoretical models above considering the following setups:

Single downstream task awareness:

of HR arrhythmia classification or R-R peak extraction. Fig. 4 shows that Dynamic-Deep follows the trends of the theoretical method successfully. Increasing the upper bound error increases the CG and the effective task loss and vice versa. For every configured upper bound DynamicDeep results with a lower effective task loss than the configured upper bound. Lastly, there is an improvement in CG for both tasks when increasing the number of compression levels from 2 to 3 (more details in [10]).

Multiple downstream tasks' awareness:

of both R-R peak extraction and HR arrhythmia classification. Supporting multiple downstream tasks introduces a tradeoff on downstream task optimization. Dynamic-Deep uses a downstream task weighted error (see eq. 2), which can be viewed as a single task awareness. This setup achieves similar results as single downstream task awareness (see tech report for demonstrations [10]). Higher CG than 64 improved the theoretical feedback-aware up to an average CG of 100, however, Dynamic-Deep had low performance utilization of those levels (Full experimental results in tech report [10]).

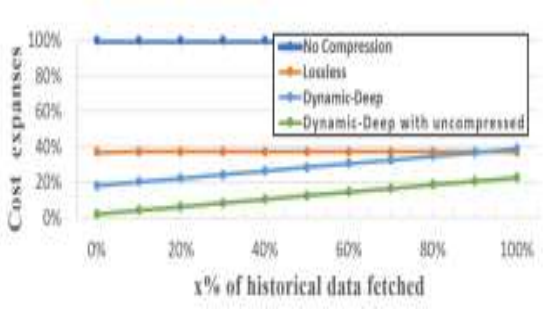


Fig. 3: Yearly cost expenses compression comparison

Cloud Cost Reduction Analysis Using Dynamic-Deep

Optimizing CG in IoT settings has a direct impact on reducing storage costs and networking bandwidth. For Cloud costs evaluation, we consider storage and computation (to account data decompression) costs since Cloud inbound traffic is usually free. We compare the following operational models:

1) Dynamic-Deep: IoT device sends a compressed representation to the Cloud side, where it is stored, or decompressed to allow downstream tasks' execution.

Dynamic-Deep with uncompressed: IoT device sends both compressed and uncompressed representations.

The compressed representation is stored while downstream tasks operate on the uncompressed representation. We assume that a domain expert reviews some portion of x% of historical sensor data, and accounts for the corresponding overhead, the cost of fetching data from storage and decompressing it, in both models. We ran these two models on Google Cloud Platform [11] using ECG data segment traffic equivalent to a small-mid hospital (200 beds). We configured the upper bound error of HR arrhythmia classification to be 0.75 and received an average CG of 48.31. We measured the computation expenses of our setup on an N1-Custom instance with 1 CPU, 2GB RAM, Intel Xeon 2.2GHz. Fig. 5 presents the measured results. Lossless compression reduces expenses by 63% regardless of the specific architecture due to its low computation usage. Dynamic-Deep with uncompressed architecture saves up to 97% cost expenses compared to no compression solution and is more efficient than lossless even with 100% data fetching.

CONCLUSION

We provided a self-tailored lossy compression structure appropriate for IoT networks that permit tuning ECG downstream duties` performances and optimize the procedure of compacting ECG statistics segments the usage of a variable-charge compression. The IoT tool learns to expect the downstream challenge blunders to permit in structured capability from Cloud services. We efficiently confirmed CG enhancements on styles of downstream duties in opposition to a SOTA CAE. Additionally we confirmed the technique permits the practitioner to stability among favored overall performance and compression benefit as a result additionally controlling Cloud costs. Future paintings will amplify the implementations to different domains. Acknowledgment - We thank Guy Vinograd from bio-T for his comments, and deeply thankful to Elad Levy for his treasured comments on version layout and analysis.

REFERENCES

[1] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," IEEE transactions on Computers, vol. 100, no. 1, pp. 90–93, 1974.

[2] M. Akbari, J. Liang, J. Han, and C. Tu, "Learned variable-rate image compression with residual divisive normalization," in 2020 IEEE International Conference on Multimedia and Expo (ICME). IEEE, 2020, pp. 1–6.

[3] J. Chen, S. Itoh, and T. Hashimoto, "Ecg data compression by using wavelet transform," IEICE TRANSACTIONS on Information and Systems, vol. 76, no. 12, pp. 1454–1461, 1993.

[4] A. F. Cheng, S. E. Hawkins III, L. Nguyen, C. A. Monaco, and G. G. Seagrave, "Data compression using chebyshev transform," 2007.

[5] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," arXiv preprint arXiv:1710.09282, 2017.

[6] Y. Choi, M. El-Khamy, and J. Lee, "Variable rate deep image compression with a conditional autoencoder," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 3146–3154.

[7] G. D. Clifford, I. Silva, B. Moody, Q. Li, D. Kella, A. Shahin, T. Kooistra, D. Perry, and R. G. Mark, "The physionet/computing in cardiology challenge 2015: reducing false arrhythmia alarms in the icu," in 2015 Computing in Cardiology Conference (CinC). IEEE, 2015, pp. 273–276.

[8] H. Djelouat, A. Amira, and F. Bensaali, "Compressive sensing-based iot applications: A review," Journal of Sensor and Actuator Networks, vol. 7, no. 4, p. 45, 2018.

[9] O. Dovrat, I. Lang, and S. Avidan, "Learning to sample," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2760–2769.

[10] A. B.-B. Elad Wasserstein, Eli Brosh, "Dynamic-deep tech-report," 2022. [Online]. Available: https://cutt.ly/bvekoKD

[11] Google. Machine types,compute engine documentation, google cloud. [Online]. Available: cloud.google.com/compute/docs/machine-types

[12] R. Goroshin and Y. LeCun, "Saturating auto-encoders," arXiv preprint arXiv:1301.3577, 2013.

[13] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," arXiv preprint arXiv:1510.00149, 2015.

[14] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," science, vol. 313, no. 5786, pp. 504–507, 2006.

[15] S. Hong, Y. Zhou, J. Shang, C. Xiao, and J. Sun, "Opportunities and challenges of deep learning methods for electrocardiogram data: A systematic review," Computers in Biology and Medicine, p. 103801, 2020.

[16] A. Koski, "Lossless ecg encoding," Computer Methods and Programs in Biomedicine, vol. 52, no. 1, pp. 23–33, 1997.

[17] R. Mark and G. Moody, "Mit-bih arrhythmia database directory," Cambridge: Massachusetts Institute of Technology, 1988.

[18] R. Pinkham, T. Schmidt, and A. Berkovich, "Algorithm-aware neural network based image compression for high-speed imaging," in 2020 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR). IEEE, 2020, pp. 196–199.

[19] P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, C. Bourn, and A. Y. Ng, "Cardiologist-level arrhythmia detection with convolutional neural networks," arXiv preprint arXiv:1707.01836, 2017.

[20] B. S. Reddy and I. Murthy, "Ecg data compression using fourier descriptors," IEEE Transactions on Biomedical Engineering, no. 4, pp. 428–434, 1986.

[21] A. Ukil, S. Bandyopadhyay, and A. Pal, "Iot data compression: Sensoragnostic approach," in 2015 Data Compression Conference. IEEE, 2015, pp. 303–312.