RDAS: A Low Latency and High Throughput Raw Data Engine for Machine Learning Systems

Anonymous Author(s)
Affiliation
Address
email

Abstract

In the era of large pretrained models, a key challenge in deep learning is the underutilization of fine-grained raw data, often replaced by information-lossy normalized data. To bridge this gap, we introduce the Raw Data Aggregation System for Machine Learning (RDAS). RDAS offers a seamless data interface, enabling machine learning systems to directly access unstructured, high-resolution raw event data with minimal latency. At the heart of RDAS lies the Message Book Model, an innovative data representation framework that underpins the system's ability to handle event data at nanosecond precision. RDAS is structured around three conceptual layers: (i) the Message Layer, featuring dual message aggregators for sequential and random access, which compile raw messages into timestamp-specific message book snapshots; (ii) the Feature Layer, which derives user-specified data features from the message book for any given moment; and (iii) the Verification Layer, tasked with real-time error monitoring and integrity assurance of the message book. A C++ implementation of these layers ensures RDAS's exceptional performance. To validate its effectiveness, we applied RDAS in an Internet of Things (IoT) scenario, demonstrating significant performance enhancements over existing methods in terms of data throughput and latency. Our results underscore RDAS's potential to revolutionize data processing in machine learning, offering a pathway to leverage the full spectrum of raw data's granularity and richness.

1 Introduction

Recent advancements in machine learning (ML) have seen large pretrained models, such as those used in natural language processing (NLP), computer vision (CV), and multi-modality fields, achieve unparalleled performance. A key characteristic of these models is their substantial data requirements. For example, ChatGPT, a model renowned for its capabilities, was trained on an extensive dataset comprising 570GB of text data from the Internet. The effectiveness of these models is further enhanced by Transformer-based architectures, which are known for scaling efficiently with increased data size [6]. Given the ongoing trend towards larger models, it is reasonable to anticipate that current data scales will continue to grow to meet these evolving requirements.

While the success of large pretrained models in NLP and CV is noteworthy, their expansion into other domains like time series prediction [9] and DNA understanding [5, 28] presents new challenges. The key issue lies in the limited quantity and granularity of training data available in these fields. For example, popular datasets in time series prediction, such as ETTh1, ETTh2, ETTm1, ETTm2, ILI, Traffic, and ECL, are relatively small, typically under 500MB, with only a few reaching between 1 to 10GB. These datasets, predominantly normalized and simplified from their original, more complex forms, result in significant information loss. ETTh1, for instance, is an hourly electricity dataset derived from higher-frequency sensor data. This loss is a major drawback, as raw datasets, often unstructured like event messages, contain richer details than their normalized counterparts.

To address this, we propose an integrated approach encompassing end-to-end data acquisition and processing. This method differs from traditional practices by processing raw data into structured form dynamically during model training. Unlike the static nature of preprocessed, normalized data, this dynamic approach allows for flexible and adaptive data transformation. This could include batch-specific normalization adjustments based on prior training results or on-the-fly data augmentations. Ultimately, this end-to-end process aims to harness the full potential of raw data, preserving its fine-grained nature for more effective model training.

However, the advantage of raw data's granularity comes with its own set of challenges, notably its unstructured nature and inherent heterogeneity. This heterogeneity significantly complicates the data acquisition and processing. Take autonomous driving systems as an example: they rely on a diverse array of sensors, including laser, image, inertial measurement units (IMUs), and odometry sensors [14]. Each sensor type generates data streams that vary in format, type, and resolution, adding layers of complexity [7] [8] [17] [10] [26].

To effectively manage the complexities of raw data acquisition and processing, thereby enabling ML systems to access the most detailed information in raw data, we propose abstracting diverse systems into a unified framework. This model conceptualizes the process as a competition among heterogeneous entities for a variety of resources, governed by multiple constraints like time of arrival, importance, ranking, cost, and gain. An entity could be, for example, a robot awaiting a task or a request for computational resources in a cluster. We envision this as N connected queues, where N represents the number of constraints, and refer to it as a 'message book' for clarity.

The message book abstraction offers several advantages for complex systems management. First, its intuitive nature facilitates easy understanding and implementation. Second, it is a versatile abstraction, applicable across various data types and systems in different domains. For example, in cloud computing, the message book can represent tasks awaiting execution, with computational power and time as the contested resources and submission time and task priority as constraints. The primary goal in such a system is maximizing the allocation of computational resources over time. This concept extends beyond cloud computing to encompass IoT systems, single-robot systems, and large-scale decentralized robotics systems. Third, the message book's construction is driven purely by data, relying on the most fundamental and unprocessed message data, thereby eliminating the need for complex data preprocessing. These attributes render the message book abstraction particularly well-suited for creating efficient, low-latency data visualization tools and data access interfaces in ML systems.

The implementation of a message book data structure is crucial for efficiently managing complex systems. Firstly, without such a model, it becomes challenging to determine the order in which participants access resources, as this decision relies heavily on various constraint functions. Secondly, given the heterogeneity, volume, and dispersed nature of the data, directly analyzing every minute activity of each participant is impractical. The message book addresses this by offering a unified representation that consolidates this diverse, granular data in real-time. This occurs concurrently with the data loading and model training processes in a ML system. As a result, the message book not only simplifies the understanding of the system's real-time dynamics but also serves as an effective intermediary for subsequent tasks. These tasks can range from visualizing system statistics to aiding in model training, thereby providing a foundational tool for various downstream applications.

Constructing the message book requires the system to adeptly route, filter, process, and aggregate all the activity information generated during its operation. We conceptualize each activity within the system as an event, with each occurring event represented as a message initiation and transmission process. Given that the message book's state is continuously evolving with the system's operation, it is crucial to model both the message book and the event message manipulations in a streaming fashion. The core research challenge we address involves developing an event-based data engine. This engine is designed to intelligently aggregate the most detailed event messages into a dynamic, general-purpose message book data structure, operating in a streaming manner. The proposed data engine is tailored to facilitate both real-time and historical data retrieval, offering efficient space and time complexity.

We organize the rest of the paper as follows. Section 2 mentioned prior work related to data processing systems, event processing systems, etc. Section 3 demonstrates the system architecture and implementation of RDAS. Section 4 shows experiments and results. Section 5 is the conclusion.

2 Related Work

Data loading frameworks Common data loading frameworks such as PyTorch Datasets & DataLoaders [1] and Tensorflow tf.data [2] are in a different position in the ML pipeline. They are for loading structured data from permanent storage such as local disk and remote storage such as S3. RDAS is to address the first-mile problem of how to acquire and aggregate raw data from data sources into a structured representation. These data loading frameworks can be naturally the next component that connects to RDAS in the ML pipeline.

Data processing system. General-purpose data processing systems include Pandas [11], Dask [20], Numpy [15]. However, they are for structured data that is generated by processing the raw unstructured data. There are also some big data frameworks that can process raw data such as Hadoop [24], Flink [1], Spark [27], Storm [2] and Hive [23]. But, they all serve as basic infrastructures and building blocks to construct data processing pipelines. They lack the higher level design to meaningfully process the raw data. This is where RDAS steps in.

Event processing system. In an IoT system, information sharing and communication are often modeled as the distribution of real-time event messages. [19] proposes an event processing solution to detect vehicle speed violations. [12] proposes an event processing engine to process events from data streams for supply chain management purposes. [22] proposes an event processing architecture to monitor the automotive manufacturing process. However, all these solutions are domain-specific. They are not a generic framework that is suitable for various kinds of domains in ML.

3 Methodology

3.1 Message Book

To maintain a simple and efficient data structure, we use queues to implement the message book data structure. If there are N constraints, there are N different queues. Each constraint corresponds to one queue and we store the specific entity objects in the queue whose constraint has the lowest priority. The entire message book data structure is a hierarchical structure. Each element of a queue with a specific constraint is a queue whose constraint is of a lower priority. For example, Figure 1 shows a minimal example of the message book. In this example, there are two constraints and constraint 1 has a higher priority than constraint 2, meaning that constraint 1 has to be met first. To be more intuitive, taking game matching as an example, constraint 1 could be the players' skill levels and constraint 2 could be the players' geographical distance. Each entity object stored in the message book represents a player in this case. Game matching for a player needs to first find other players with the same skill levels then among which find the player with the smallest distance. In this case, each element of the constraint 1 queue is a Struct that includes some basic information for that level such as the total number of players that are in that level, and a pointer to the queue of players in that level. In the queue of players, each element is a Struct representing a player and they are sorted according to constraint 2, which is the geographical distance that is of a lower priority. We use C++ standard library to implement the message book. The details are in **??**.

Traverse The traversing of all entities (players) follows the priority of the constraints that are level first and entity second. This process starts from the index of the first level. For each level, it starts from the first entity and traverses all entities following the pointer to the next entity in each entity's entry. Then, it follows the pointer to the next level in the current level's entry and traverses all entities of the next level. This process goes on until it traverses all levels' entities. The traversing has $O(n)$ time complexity.

Random access to existing entities Each time a new level or a new entity is added to the message book, their id-to-index mapping is saved in the hash map. Thus, for the random access to any specific entity or level, it is $O(1)$ time complexity using the hash map.

[1] `https://pytorch.org/tutorials/beginner/basics/data_tutorial.html`
[2] `https://www.tensorflow.org/guide/data`

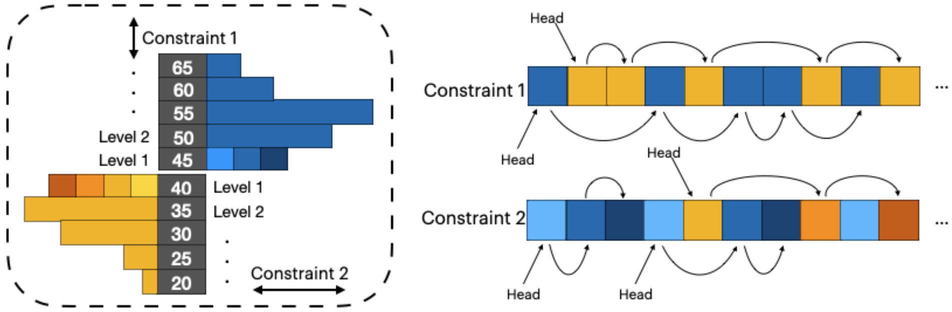

Figure 1: A Minimal example of the message book implementation. Left: a visualization of the message book with two constraints. Constraint 1 has a higher priority than constraint 2. Right: the implementation of the message book using two queues. The top corresponds to constraint 1 and the bottom corresponds to constraint 2. Only the queue of the least prioritized constraint(bottom queue) stores the entity objects. The top queue only stores level information.

Add entities or levels To add a new entity to the queue of Constraint 1 or add a new level to the queue of Constraint 2, the algorithm first gets the index of free entry in the queue by popping an element from the corresponding FreeVector. Then, it creates a Struct representing the new entity or new level in the location indicated by the index in the corresponding queue. This is $O(1)$ time complexity.

Delete entities or levels To delete an entity or a level, RDAS gets its index in the corresponding queue using the corresponding hash map. Then, it pushes that index into the corresponding FreeVector. This is $O(1)$ time complexity.

3.2 Data Transmission Format and Channels

The domains of machine learning systems that our system is for are those whose raw data are event data. For instance, in transportation systems, all raw data are event data that depict vehicles', road sensors', and traffic lights' status at each time point. In cloud systems, all the raw data are event data about different machines, different components, or different software of the system. In IoT or robotics, all the raw data are the events generated by different sensors, devices, or robots. Because of the blooming of edge device computation power and high-speed, low-latency communication technologies, highly decentralized IoT systems and robotics systems are becoming a reality. We choose a data format that is suitable to the current and future trends of heterogeneous event data transmission scenarios in various domains. In this case, we need a suitable data storage and transmission format that is capable of providing the properties of large volume, high frequency, low latency and high compatibility with network transmission of the event messages data because all the sensors and other system participants transmit data over the network [4] [16] [25] [13]. In RDAS, we store all the raw event messages in *pcap* (Packet Capture) format. Pcap is a direct capture of the data packet (event messages) that the senders send over the network [21]. It provides a single truth of source and an unbeatable granularity of up to nanosecond regarding the timestamp [3]. These features ensure that it has high credibility and enough capacity for any highly heterogeneous raw data source requiring various degrees of transmission latency, transmission frequency, and communication credibility.

Besides the foundational data storage format for the raw event message data, we also define a concept called data transmission channel in RDAS. There are four different channels as shown by Figure 2, which are actually four parallel data feeds when the system is operating. They are Alpha channel 1, Alpha channel 2, Beta channel 1 and Beta channel 2. Alpha channel only includes incremental information so that the transmission can have extreme compactness and high frequency. The Alpha channel is necessary when the frequency of new updates is very high because it is impossible to store the snapshot of the entire message book's status for every new update regarding the scale of the storage that is needed and the large percentage of redundant data in each snapshot. Beta channel only includes snapshot information the sender sends at a slightly lower frequency so that the transmission data volume is not too large. In the meantime, any device listening to this feed at any time can get a relatively new snapshot of the sender's state and start evolving the state from this snapshot using information from the Alpha channel. To ensure a high degree of robustness of the data transmission, both Alpha channel and Beta channel is a set of two parallel channels. Senders send the same packets simultaneously on channel 1 and channel 2. Receivers can use channel 1 and channel 2 to do cross verification or if some packets are corrupted on one channel, the system can recover them from the other channel. It is also noteworthy that the use of Alpha channel and Beta channel is flexible. For senders that only need to send stateless information, the use of Beta channel is optional. For senders that have a very limited amount of state data and have no requirement on high-frequency transmission, they can choose to only use the Beta channel to broadcast their full state periodically or whenever there is an update. Such a high degree of flexibility is an essential property that makes RDAS suitable to highly heterogeneous systems.

3.3 Core Data Processing Pipeline

The data processing pipeline of RDAS consists of four phases: channel merging, channel routing, channel decoding, knowledge distilling, and verification. The first three phases support the message layer. The last phase supports the feature layer and the verification layer. All these phases are piped to each other in a streaming manner, which means that it is not the case where a phase first processes all the data and then hands them over all at once to the next phase. It is the alternative case where when each phase processes a very small piece of the data, it immediately hands it over to the next phase and then starts processing the next piece. It is easy to see that in this streaming manner, all the events can be processed in the same chronological order as in that they took place and the large data size induced along the temporal dimension does not impair the processing speed.

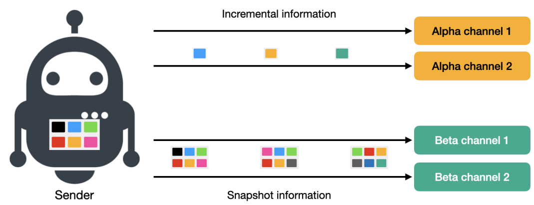

Figure 2: Two types of data channels: Alpha for high-frequency incremental information; Beta for low-frequency snapshot information. Each channel has a backup channel for robustness.

In the first phase, channel merging, the system merges all channels from all senders together into one monolithic stream. This process makes sure all the network packets in this stream are in chronological order using their timestamp of nanosecond granularity. In the second phase, channel routing, the system routes each packet to a specialized channel decoder that is responsible for decoding the data of a specific channel. The routing is based on the unique IP address of the sender. The third phase, channel decoding, as shown by Figure 3 is a major part of the pipeline and handled by a channel decoder. There is a recovery mode and an incremental mode of the channel decoder and these two modes can switch back and forth into each other. It is usually in the recovery mode when the decoder is at the start of the decoding process or the decoded stream has corruptions for some reason in the middle of the decoding process and the decoder needs to be re-calibrate it to a correct checkpoint of the states of the sender. The incremental mode is to apply incremental changes to a base snapshot of states so that the system can maintain an evolving and always on-sync message book. There are two sub-phases in the decoder. We call the one before the decoder starts processing the packet on-packet-start, which is an API allowing the user to define and conduct any task before the decoder processes the packet. RDAS generates message book snapshots through this API. We call another one after the decoder processes the packet on-packet-end. It is also an API to allow users to define and conduct any task before the channel decoding phase ends for this specific packet.

When the channel decoder is in recovery mode and a packet comes in, after the on-packet-start phase, the decoder checks the packet if it is from an Alpha channel or a Beta channel. If it is from an Alpha channel, the decoder would push it into the buffer queue without processing it. If it is from a Beta channel, the decoder hands it to a message parser that parses the content of the packet and produces a set of atomic operations. An atomic operation is a standardized and generic operation whose definition is a function. For example, some of the atomic operations are `Add Event`, `Modify Event`, and `Cancel Event`. The data engine uses a single set of available atomic operations of limited size for all kinds of data. Then, the decoder applies the set of atomic operations to the message book data structure so that the state of the message book incorporates the new information from the newcomer packet. Usually, in the recovery mode, because the message book is either in an empty state or corrupted state, the atomic operations are just `Add Events` to fill the empty message book or overwrite the existing message book.

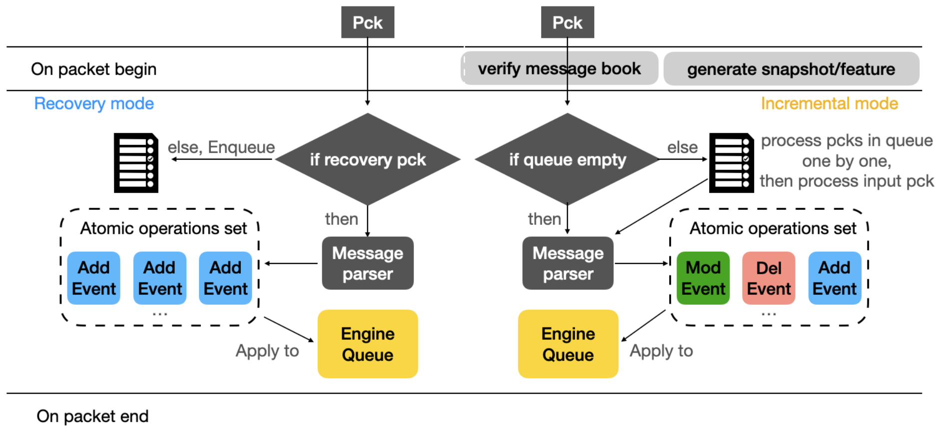

Figure 3: The feed decoding process: (1) two modes, recovery mode, and incremental mode. In recovery mode, the system uses the messages from the Beta channel to recover a snapshot of the message book from scratch. In the incremental model, the system applies incremental messages to the snapshot to evolve the message book. (2) The system preserves two user APIs on-packet-begin and on-packet-end, to generate useful information on the fly such as snapshots and features and do verification at any timestamp.

When the channel decoder is in incremental mode, the process is almost the same as the recovery mode. The decoder first checks the buffer queue. If it is not empty, the decoder processes packets in the queue one by one until the queue is empty. After emptying the buffer queue, the decoder starts to handle the input packet. If the input packet is from a Beta channel, the decoder discards it. If it is from an Alpha channel, the message parser parses it to a set of atomic operations then the decoder applies them to the message book to incorporate new information from the packet.

The last phase in the pipeline is knowledge distillation and verification. It supports both the feature layer and the verification layer. This phase usually does not happen at the end of decoding the current packet. It happens at the start of decoding the next packet to guarantee the captured snapshot is the closest timestamp to the boundary timestamp. The feature generation and message book verification happen through the on-packet-begin API so that users have the flexibility to tailor these two processes. This phase is an essential bridge users can use to connect the message book to various sorts of downstream tasks such as data visualization terminals, analytical models, logging devices, etc.

3.4 Time Traveling

The time-traveling feature is a representative feature of RDAS based on its core data processing pipeline. It allows the query of a snapshot of the data stream at any timestamp with a constant maximum loading time no matter how far the snapshot's timestamp is from the start of the stream. The mechanism behind the time traveler is that the system first generates and caches a bunch of snapshots in the permanent storage at a frequency much lower than the original raw data granularity, for instance, every 10 minutes. Then, when the user jumps to any specific timestamp, RDAS first loads the snapshot from the cache that is the closest to that timestamp and then applies all incremental messages between the snapshot timestamp and the target timestamp to the message book to generate the target snapshot the query requests for.

4 Experiment

4.1 Case Study

We demonstrate the efficacy of RDAS by building a simulated high-frequency robotics logistics system and using it to acquire and parse the unstructured raw event data into the message book representation that is ready to be connected with the later data loading and model training pipeline. The reason that we choose to simulate a scenario in such a domain is that its raw data has high heterogeneity and high granularity, which provides enough complexities to test our system. Besides, the logistics industry is one of the kinds of complex systems that foresees to eventually transition from a labor-intensive industry into an automation industry. In 2021, there are about 1.3 million delivery drivers and about 40 million packages are in delivery every day in the US. In China, those numbers are about 4 million and 3 billion. However, this industry receives many critiques from both workers about its intense working schedule and clients about its far-from-satisfaction efficiency. Many delivery companies are considering or attempting to automate the delivery process by adopting delivery robots.

In an efficient logistics system with all delivery drivers as delivery robots, millions of robots should be accurately and optimally dispatched to deliver tens of millions of packages based on their capacity and package pick-up distance and deliver them to the client. It is essentially a message book model with very high precision in timestamps and large enough capacity to handle millions of events sent from different robots in both real-time and offline analysis.

In this scenario, there are two types of participants in the system. The first one is the local package dispatch centers that receive packages from higher-level dispatch centers from time to time. Whenever a batch of packages arrives at a local dispatch center, it sends out a Ready-to-Give (RtG) request to the system. The second one is the delivery robots that send a Ready-to-Take (RtT) request associated with the local dispatch center that is the nearest to them anytime they have capacity and are ready to fetch and deliver some new packages from the local dispatch center. During the operation, the system maintains a message book for each local dispatch center in parallel. In the message book, the constraints that decide the order of events are the distance and the time of arrival (ToA) of the request. The distance is discrete and represents the straight line distance to that specific dispatch center. Thus, there will be two types of distances. The first type is the distance between the destination of the delivery and the dispatch center. The second is the distance between the delivery robot and the dispatch center. There are different zones for different distance ranges. For instance, `Zone 1` is within [0km, 3km), `Zone 2` is within [3km, 6km), `Zone 3` is within [6km, 9km), etc.

During the operation of the system, all the delivery robots keep sending location/distance information periodically (for example, every 10 min) to the local dispatch center that they belong to. They are all in the message book no matter whether they are on their way to deliver packages or to the dispatch center to fetch new packages. Besides, all the packages are in the message book until the delivery is complete. They also have two states in the message book, either `being-delivered` or `to-be-delivered`. Once the dispatch center assigns a package to a delivery robot, the package's state changes from `to-be-delivered` to `being-delivered` and it leaves the message book once the delivery is complete.

4.2 Experiment Setup

The experiment is twofold. One is to demonstrate RDAS's large throughput, the other is to show the low latency and the low space cost of RDAS's time-traveling feature. We use ROS to simulate the dynamics of the aforementioned logistics system and generate the event message data. We assume there are always 2000 delivery robots and 20000 packages on the message book and the robots and the dispatch center send messages at some preset frequency. We built a separate ROS program to uniformly sample and generate these messages from all the possible message types to simulate reality. There are in total six types of event messages that affect the message book status. Three are for delivery robots and the other three are for packages. For each data stream, the system generates a snapshot every 10 minutes and caches it to the disk.

For delivery robots, the first type of message is `AddRobot`. When a new robot becomes online, the robot sends this message. Its fields include `distance`, `capacity`, `state`, `last_update_time`. The second type is `ModifyRobot`. When a robot's status changes between `Occupied` and `Empty`, it sends out this message. The third type of message is `RemoveRobot`, when a robot becomes offline, it sends this message. For packages, when a new package arrives at the dispatch center and waits for delivery, the dispatch center sends out an `AddPackage` message. When a package's status changes from `to-be-delivered` to `being-delivered`, the dispatch center sends an `ModifyPackage` message. Lastly, when a package delivery is complete, the dispatch center sends a `RemovePackage` message to remove the delivered package from the queue.

For the throughput experiment, the baseline system is the same as RDAS except that the baseline system does not have the incremental message aggregating mechanism and message book representation. Instead, it assumes each message includes the full picture. We assume there are two message data streams for RDAS and the baseline system respectively. In each time unit, two streams have the same number of messages and the same amount of information such as the number of robots and packages and their status. The only difference is that one stream uses incremental messages and the other uses full-picture messages. We measure the total size of the messages each system processes within one second. Under the same network bandwidth, a system processes more information (higher throughput) at each time unit if it requires a smaller total message size to recover the same amount of information. Thus, we represent the throughput by taking the inverse of the total message size per second with a scale.

The time-traveling feature is a representative feature that can demonstrate the superb performance of RDAS. We conduct experiments to measure the latency and the space cost of the time-traveling mechanism. To have a low latency for this mechanism, there are two types of overhead to consider. The first is the time cost to generate the snapshot cache. The second is the latency to apply the incremental messages between the loaded snapshot time and the target timestamp to generate a new snapshot on the fly. We measure both of them. Furthermore, we also measure the size of disk space to store the snapshot cache. The experiment demonstrates that the random access time traveler enables an O(1) latency with respect to the number of minutes away from the initialization point of the data stream and the size of the disk space to save the snapshot cache is within a reasonable range.

4.3 Results

We can see from Table 1 and Figure 4 that with the incremental message aggregation and the message book representation, RDAS's throughput is several magnitudes higher than the baseline system. When the data stream's frequency is at 10^3hz, RDAS's throughput is about 6 times larger than the baseline. The discrepancy keeps increasing when the stream frequency increases. When the frequency is 10^7hz, RDAS's throughput is about 400 times higher than the baseline.

Table 1: Throughput of RDAS and the baseline system for data stream of different frequencies. It is a scaled value of the inverse of the total message size per second.

	Data stream frequency				
System	$\mathbf{10^3}$**hz**	$\mathbf{10^4}$**hz**	$\mathbf{10^5}$**hz**	$\mathbf{10^6}$**hz**	$\mathbf{10^7}$**hz**
RDAS	1	163.68	1850.35	5989.47	10366.29
Baseline	0.15	1.47	14.87	23.66	26.18

To demonstrate RDAS's low latency, we first run the system several times to repeatedly measure the time it takes to process 10000 packets and take an average. The final result is that it takes about 0.01 seconds to process 10000 network packets. If the data simulation program generates the data stream at a 100hz frequency, a day(24 hours) of the streamed Pcap data has 8.64 million messages and the whole snapshot cache generation for them only takes 8.64 seconds. Even the stream is of 100000hz frequency, meaning on average, there are 100000 messages every second, which is already very unlikely in real-world robotics and IoT systems or systems in other domains, RDAS only takes 2.4 hours to cache a day of snapshot data. Considering the snapshot cache generation is an offline task, such a level of time complexity is already sufficient. It is also noteworthy that in these settings, the latency for the time traveler to generate a requested snapshot on the fly has a reasonably short upper limit which is 1 second.

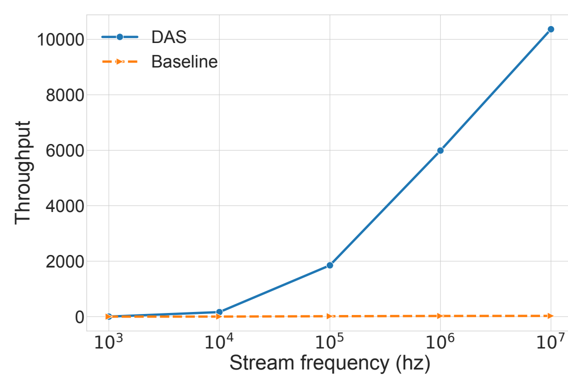

Figure 4: Throughput (the larger the better): starting from the stream frequency of 10^3hz, the throughput of RDAS is about 6 times larger than that of the baseline system. When the frequency increases to 10^7hz, the RDAS is about 400 times larger than the baseline. RDAS's throughput increases much faster than the baseline's when the frequency increases.

We also measure the random access waiting time in querying snapshots of the data stream at any timestamp to demonstrate RDAS's low latency. We construct a baseline system using the same codebase as RDAS except that the baseline does not have the random access message aggregating feature and it can only start building the target snapshot from the initialization point of the data stream. By comparing their random access waiting time, we can clearly see from Table 2 and Figure 5 that RDAS has a constant bound but the baseline keeps increasing, causing long latency when the target timestamp is far away from the initialization point.

Table 2: Random access waiting time (milliseconds) for RDAS is a piece-wise linear function with the peaks always at a similar constant number. However, it is an increasing linear function for baseline with no bounds, which indicates a much higher and eventually unacceptably long latency for the user to obtain a snapshot.

	Time distance range (min)					
System	**0**	**5**	**10**	**15**	**20**	**25**
RDAS	22.0	2886.7	23.6	2983.5	22.4	3011.8
Baseline	23.4	2874.3	6023.7	8800.9	12084.4	14858.9

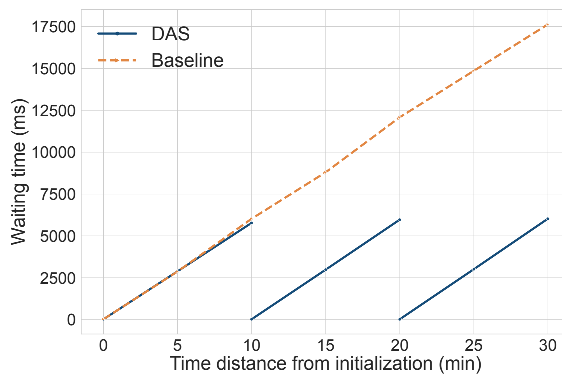

Figure 5: Random access waiting time: On the data stream of 10000hz frequency, RDAS's random access waiting time always has a nearly constant bound at around 6000ms. However, the baseline model has a linearly increasing waiting time with respect to the further and further timestamps.

Lastly, we examine the disk space consumption of the snapshot cache. Although the snapshot size varies in different systems in different domains, our logistics robotics system example, with 2000 delivery robots and 20000 packages on the message book for each dispatch center at any time, is already ambitious and thus, representative enough. We assume that without compromising the user experience, a 1-second latency (waiting time) is a reasonable upper limit when the user jumps around to query snapshots at random timestamps. It means we need to at least cache 1 snapshot for every 1 million messages because the processing speed is 1 million messages per second. Hence, disk consumption is a meaningful and critical metric. We generate a day of snapshot cache by generating one snapshot for every 1 million messages. The average size of each snapshot file is 100KB. Following the same deduction as above, if the simulated data stream is of 100hz frequency, the snapshot cache of one day is only about 860KB and it is about 860MB if the frequency is 100000hz. This shows that the space cost of RDAS is very low.

5 Conclusion

We propose RDAS, a general-purpose raw data acquisition and processing system for machine learning systems in various domains. It provides a unified data interface to bridge the unstructured, high-resolution and heterogeneous raw data of up to nanosecond granularity to common data loading and model training pipeline. It features a novel data representation mechanism, the message book, with the incremental message aggregation mechanism and a low-latency random access time traveler. RDAS allows users to quickly query a snapshot of the message book at an arbitrary timestamp. The experiments demonstrate RDAS has a high throughput, low latency, and consumes reasonably small disk space with full support to high-resolution raw event data. Future works could provide a programming language agnostic interface that enables quick integration into any existing machine learning frameworks.

References

[1] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas. Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 36(4), 2015.

[2] R. Evans. Apache storm, a hands on tutorial. In *2015 IEEE International Conference on Cloud Engineering*, pages 2–2. IEEE, 2015.

[3] F. Girela-Lopez, E. Ros, and J. Diaz. Precise network time monitoring: Picosecond-level packet timestamping for fintech networks. *IEEE Access*, 9:40274–40285, 2021.

[4] H. Huang and A. V. Savkin. Towards the internet of flying robots: A survey. *Sensors*, 18(11):4038, 2018.

[5] Y. Ji, Z. Zhou, H. Liu, and R. V. Davuluri. Dnabert: pre-trained bidirectional encoder representations from transformers model for dna-language in genome. *Bioinformatics*, 37(15):2112–2120, 2021.

[6] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[7] T. Kim, S. Lee, T. Hong, G. Shin, T. Kim, and Y.-L. Park. Heterogeneous sensing in a multifunctional soft sensor for human-robot interfaces. *Science robotics*, 5(49):eabc6878, 2020.

[8] B. Liu, L. Wang, M. Liu, and C.-Z. Xu. Federated imitation learning: A novel framework for cloud robotic systems with heterogeneous sensor data. *IEEE Robotics and Automation Letters*, 5(2):3509–3516, 2020.

[9] Q. Ma, Z. Liu, Z. Zheng, Z. Huang, S. Zhu, Z. Yu, and J. T. Kwok. A survey on time-series pre-trained models. *arXiv preprint arXiv:2305.10716*, 2023.

[10] S. Manjanna, A. Q. Li, R. N. Smith, I. Rekleitis, and G. Dudek. Heterogeneous multi-robot system for exploration and strategic water sampling. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4873–4880, 2018.

[11] W. McKinney et al. pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9):1–9, 2011.

[12] F. Nawaz, N. K. Janjua, and O. K. Hussain. Perceptus: Predictive complex event processing and reasoning for iot-enabled supply chain. *Knowledge-Based Systems*, 180:133–146, 2019.

[13] T. Nestmeyer, P. Robuffo Giordano, H. H. Bülthoff, and A. Franchi. Decentralized simultaneous multi-target exploration using a connected network of multiple robots. *Autonomous robots*, 41(4):989–1011, 2017.

[14] Y. Nitta, S. Tamura, and H. Takase. A study on introducing fpga to ros based autonomous driving system. In *2018 International Conference on Field-Programmable Technology (FPT)*, pages 421–424. IEEE, 2018.

[15] T. E. Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[16] A. A. Pereira, J. P. Espada, R. G. Crespo, and S. R. Aguilar. Platform for controlling and getting data from network connected drones in indoor environments. *Future Generation Computer Systems*, 92:656–662, 2019.

[17] J. P. Queralta and T. Westerlund. Blockchain-powered collaboration in heterogeneous swarms of robots. *arXiv preprint arXiv:1912.01711*, 2019.

[18] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[19] S. Rakkesh, A. Weerasinghe, and R. Ranasinghe. Simulation of real-time vehicle speed violation detection using complex event processing. In *2016 IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*, pages 1–6, 2016.

[20] M. Rocklin. Dask: Parallel computation with blocked algorithms and task scheduling. In *Proceedings of the 14th python in science conference*, volume 130, page 136. Citeseer, 2015.

[21] L. F. Sikos. Packet analysis for network forensics: A comprehensive survey. *Forensic Science International: Digital Investigation*, 32:200892, 2020.

[22] M. Syafrudin, G. Alfian, N. L. Fitriyani, and J. Rhee. Performance analysis of iot-based sensor, big data processing, and machine learning model for real-time monitoring system in automotive manufacturing. *Sensors*, 18(9):2946, 2018.

[23] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2):1626–1629, 2009.

[24] T. White. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

[25] A. F. Winfield. Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots. In *Distributed autonomous robotic systems 4*, pages 273–282. Springer, 2000.

[26] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

[27] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. In *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10)*, 2010.

[28] Z. Zhou, Y. Ji, W. Li, P. Dutta, R. Davuluri, and H. Liu. Dnabert-2: Efficient foundation model and benchmark for multi-species genome. *arXiv preprint arXiv:2306.15006*, 2023.

[29] H. Zimmermann. Osi reference model-the iso model of architecture for open systems interconnection. *IEEE Transactions on communications*, 28(4):425–432, 1980.

A Additional Background

Robotics system. ROS (Robotics Operating System) [18] is a system providing communication layers on top of machine operating systems to a computation cluster. Heterogeneous robots can use it to communicate with each other or a centralized cloud system. However, ROS only provides an underlying communication infrastructure and it does not deal with a higher level regarding data processing, which is where RDAS can step in.

Data format. Some other file formats are popular in data-logging scenarios. For instance, Ros Bag file format [3] is the format that Ros uses to store Ros messages in files. However, it is bound to Ros's ecosystem. Another general-purpose message recording data format is the MCAP format [4]. However, both Bag and MCAP's disadvantage compared the PCAP format our data engine uses is that they capture data from the application layer while PCAP captures data from the lower network layer according to the 7-layer OSI model [29]. Capturing from the network layer is much faster. Besides, network layer has the additional network information that helps network monitor and analysis. It is crucial for the scenarios that put the requirement for low latency, high throughput and high robustness to extreme.

[3] `https://wiki.ros.org/Bags/Format/2.0`

[4] `https://foxglove.dev/blog/introducing-the-mcap-file-format`

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading “NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Contributions and scope are stated clearly in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations on the theoretical and empirical results are discussed in their respective sections, as well as Section **??**.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The experiment information needed to reproduce the main experimental results are in Section **??** and Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Code is not available at the time of submission but will be available as an open-source repository upon acceptance.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: They are specified in Section **??** and Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: See Section **??**

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: They are in Section **??** and Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed and followed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This work is about methodological and foundational development, which we do not see any societal impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Data and models used are all cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: Code is not available at the time of submission but will be available as an open-source repository upon acceptance.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.