DeepSQA: Understanding Sensor Data via Question Answering

Tianwei Xing
University of California, Los Angeles
twxing@ucla.edu

Luis Garcia
University of Southern California ISI
lgarcia@isi.edu

Federico Cerutti
University of Brescia
federico.cerutti@unibs.it

Lance Kaplan
CCDC Army Research Lab, Adelphi
lance.m.kaplan.civ@mail.mil

Alun Preece
Cardiff University
PreeceAD@cardiff.ac.uk

Mani Srivastava
University of California, Los Angeles
mbs@ucla.edu

ABSTRACT

The ubiquity of mobile, wearable, and IoT devices enhances humans with a network of environmental sensors. These devices capture raw, time-series measurements of scalar physical phenomena. To transform the data into human-digestible representations, deep learning methods have enabled high-level interpretations of the opaque raw sensory data. However, interfacing models with humans requires flexibility to support the vast database of human inquiries about sensor data. Deep learning models are usually trained to perform fixed tasks, limiting the inference outputs to a predefined set of high-level labels.

To enable flexible inference, we introduce DeepSQA, a generalized Sensory Question Answering (SQA) framework that aims to enable natural language questions about raw sensory data in distributed and heterogeneous IoT networks. Given a sensory data context and a natural language question about the data, the task is to provide an accurate natural language answer. In addition to the DeepSQA, we create SQA-Gen, a software framework for generating SQA datasets using labeled source sensory data, and also generate OppQA with SQA-Gen for benchmarking different SQA models. We evaluate DeepSQA across several state-of-the-art QA models and lay the foundation and challenges for future SQA research. We further provide open-source implementations of the framework, the dataset generation tool, and access to the generated dataset, to help facilitate research on the SQA problem.

CCS CONCEPTS

• **Computing methodologies → Machine learning**; **Neural networks**; **Natural language processing**; • **Computer systems organization → Sensor networks**; • **Human-centered computing → Human computer interaction (HCI)**.

KEYWORDS

Question answering, Neural networks, Sensor data processing

ACM Reference Format:
Tianwei Xing, Luis Garcia, Federico Cerutti, Lance Kaplan, Alun Preece, and Mani Srivastava. 2021. DeepSQA: Understanding Sensor Data via Question Answering. In *International Conference on Internet-of-Things Design and Implementation (IoTDI '21), May 18–21, 2021, Charlottesvle, VA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3450268.3453529

1 INTRODUCTION

Sensors in various embedded, wearable, and mobile IoT devices produce enormous amounts of data, which algorithms help transform into actionable insights and predictions that guide decisions and interventions at various scales. While recent years have seen the emergence of powerful deep-learning-based neural network models, capable of making rich and complex inferences from large amounts of sensory data, current data-to-decision pipelines are highly constrained as they employ specialized models for specific tasks such as detecting a set of events and activities.

Imagine instead a future where a human decision-maker is not limited to a fixed set of inferences computed from sensory data, and could instead ask flexible natural language questions about events and activities present in the data and get answers. For example, instead of a processing pipeline extracting fixed information about various activities of daily living from a user's wearable and ambient sensors, imagine the user being able to ask "How long did I exercise between lunch and dinner yesterday?" and "How many times did I drink water yesterday?" Or, imagine that instead of being presented with a fixed set of traffic events and statistics derived from time-series data from traffic sensors, a city manager could ask questions such as "How long did the congestion on Highway 99 last?" and "Was there an accident during the hour preceding the congestion?"

Our research is inspired by a vision of providing users with the ability to extract a variety of inferences from sensory data, by asking flexible natural language questions instead of being limited to the rigid outputs of a fixed set of classification and regression models. To achieve such a capability, one needs a framework that can formulate answers to novel and arbitrary questions about an underlying sensor dataset without requiring a new model to be trained for each question, and also be able to incorporate new knowledge efficiently. Recent advances in deep-learning-based natural language processing and its use for tasks such as asking questions about images[3], texts[28], and databases[11] suggest that restricted forms of such a capability are certainly possible for spatiotemporal sensor data as well. This paper presents the results of our exploration of this problem.

In this work, we propose the DeepSQA, a generalized framework to address the Sensor Question Answering (SQA) problem of answering natural language questions about raw sensory data in distributed and heterogeneous IoT networks. Figure 1 illustrates some exemplar questions that DeepSQA can solve. Suppose a user interfaces with a set of IoT devices, e.g., a smartphone and a smart

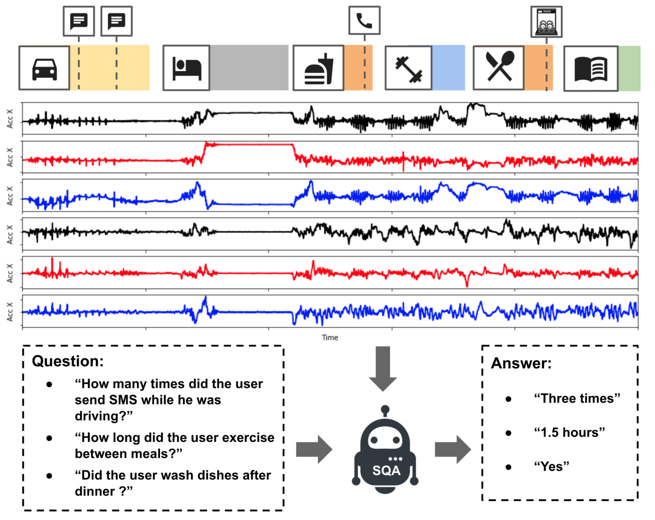
Acc X
Time
Question:
• "How many times did the user send SMS while he was driving?"
• "How long did the user exercise between meals?"
• "Did the user wash dishes after dinner ?"
SQA
Answer:
• "Three times"
• "1.5 hours"
• "Yes"

Figure 1: Examples of open-ended natural language questions supported by our proposed Sensor Question Answering model.

band to keep track of their daily activities. With multimodal data collected by wearable devices, they may want to query their behaviors from different perspectives. For example, the user could ask "How many times did I send an MSM while driving?" or "How long did I exercise between lunch and dinner?" Based on the collected sensory data, the SQA system should answer each of these questions accordingly with the correct natural language answer, e.g., "three times" or "1.5 hours".

In addition to DeepSQA, we also introduce SQA-Gen, a software framework for generating SQA datasets from underlying labeled sensory data. To evaluate DeepSQA, OppQA dataset is created with SQA-Gen for complex human activity question answering. This dataset focuses on spatial-temporal relationships across different sensors, and it contains over 1K sensory contexts and 91K questions. A detailed analysis is performed on this dataset with various modern baseline models, providing insights into the SQA task. We provide open-source implementations of the DeepSQA framework, SQA-Gen data generation tool, and access to the OppQA dataset[1]. We believe this will benefit the community of sensory question answering research.

Contributions. Our contributions are enumerated as follows.

- Firstly, we introduce the DeepSQA framework, the generalized QA framework to address the SQA problem by enabling natural language questions about raw sensory data in distributed and heterogeneous IoT networks.
- Secondly, we propose SQA-Gen, a software framework to generate SQA data using labeled source sensory datasets. Based on SQA-Gen, we create the first SQA dataset on complex human activity question answering, to benchmark SQA models' performance on natural language QA about spatial and temporal properties of raw sensory data streams.
- Thirdly, we evaluate DeepSQA across several state-of-the-art QA models on OppQA, and enumerate the challenges at the frontier of SQA.
- Lastly, We provide an open-source implementation of DeepSQA, SQA-Gen, as well as the access to the OppQA dataset.

2 RELATED WORK

In this section, we review the related works on sensory data processing and question answering in different domains.

Machine Learning for Sensory Data. Sensory data processing is not only a critical problem in the signal processing field, but also a hot topic for machine learning applications. Vibrant research has been performed in the field like visual and acoustic domains. However, in this research, we are interested in sensory data that humans cannot easily understand, i.e., sensory data from devices like inertial sensors that are nothing more than a series of scalar physical phenomena. Due to the lack of senses or a standard vocabulary, humans would have a hard time associating these time-series values with the high-level symbolic concepts.

With the advancement of machine learning techniques and compute power, it is possible to make inferences on raw sensory data using a data-driven approach. [23] presents a survey on using wearable sensory data to perform Human Activity Recognition(HAR). Recent research [10, 30, 33] shows that deep learning techniques can better infer human activities. [34] uses the WiFi signal to achieve device-free activity recognition. Besides, time-series sensory data like EEG [4, 35] or GPS [6, 39], are used to make high-level inferences about different activities and events.

However, all the existing works focus on using sensory data to perform fixed predefined tasks, e.g., classifying subject behavior or predicting user sentiment. If the tasks are changed, the data need to be re-labeled, and the models have to be retrained. Despite the inefficiency of data labeling and the cost of model training, there's still no framework that can provide solutions for arbitrary tasks within a specific range.

Question Answering. On the other hand, research in the visual and natural language processing domains propose a framework that allows task-aware inferencing, in the form of question-answering (QA). In QA, a model is trained to take both a question and its context as input, and answer the given question based on the context data. This QA task requires models to be capable of not only processing raw context data, but understanding natural language questions as well. The questions asked to the model determine the tasks that the model performs. Different tasks, based on either the same or different context data, can be handled by a single QA model. There has been a body of work on question answering applications in different domains, which can be broadly categorized into two groups based on context data.

The first group of QA tasks deals with static data, such as texts, images, charts, and structured data like tables and knowledge bases. As the main focus of this paper is on the question answering with multimodal data, we do not discuss the related work of natural language QA [28] here.

In the Visual Question Answering(VQA) domain, large datasets [3, 14, 18] have be proposed to give fair benchmarks. [18] creates a large VQA reasoning dataset using generated images and questions,

[1] 1The data and codes of DeepSQA are available at https://github.com/NESL/DeepSQA.

which helps reduce data biases and test model generalization ability. In [14], questions are synthesized on real-images to extend the domains further. Different methods are proposed to handle the VQA problem. A major trend is to fuse the image and question features in different ways, such as combining CNN features of images and LSTM features of questions together[3]. Models based on attention mechanism [22, 37] use concatenated feature to first calculate attention weights of the image, and then predict the answer based on the attended area. FiLM[27] proposes a model that interleaves standard CNN layers with linear layers, tilting the layers' activations to reflect the specifics of given questions. More recent work shows that recurrent approaches like [13] with multiple-step reasoning can achieve good performance on datasets where complex reasoning is required. With additional supervision, neural-symbolic methods show better performance on the VQA task. [12, 19] use modular approaches to synthesize models that have the same structure as the questions. [38] uses deep learning model to parse both the image scenes and questions, and then answers different queries in the semantic space. Instead of using latent feature representations only, a body of work[2] solves VQA problems using additional object-level features and achieves excellent performance. However, this method could not be adapted to the SQA problem, since obtaining the semantic feature is hard for the opaque sensory data.

There are also emerging domains that ask questions on the context of other modalities, like graphs and tables. [20, 21] propose the task of question-answering on graphs and diagrams, where the answer categories are context-dependent, and the answers to questions are sensitive to small variations in the diagrams. [11] uses a BERT[9]-like structure to answer questions based on tabular data.

In summary, all of the work discussed above focus on answering questions with static data, where complex temporal reasoning is not required to get the correct answer.

The second group of QA tasks relies on time series data. [16, 17, 24, 31] solve the problem of video QA with video clips as context data. [1] introduces the task of audio question answering, and uses vision-based models to process the spectrogram features extracted from audio contexts. Although spatio-temporal reasoning is performed in these models, they can only handle context data collected from a single source. Question answering with distributed sensory data from multiple sources with heterogeneous modalities requires the ability to perform sensory fusion and inter-sensor spatial reasoning. Also, it is unclear whether these approaches discussed above are still effective on sensory data that are opaque to humans, such as the IMU data that humans cannot understand.

Semantic Parsing. An alternative approach to handle flexible natural language queries is semantic parsing[5]. Semantic parsing is the process of mapping a natural-language sentence into a logical form, which is a machine-understandable, formal representation of its meaning. After this process, the questions in logic form can be used to query structured or semi-structured knowledge bases. In our SQA problem, to enable a semantic-parsing-based system, effort must be first taken to train a model that processes and maps raw sensory data into a logical semantic space interfacing with logic queries. However, it is impossible to define such an informative semantic space for opaque and unstructured sensory data, especially when tasks are not provided yet. Also, preparing the annotated data for training the sensory model is a huge burden. Therefore, in this paper, we choose to use an end-to-end approach, which implicitly parses the natural-language questions using neural networks, to tackle the SQA problem. Related work in the semantic parsing field is not discussed here since it is beyond our scope.

3 FORMALIZING SQA PROBLEM AND DESIGN OF DEEPSQA

In this section, we formally define the problem of Sensory Question Answering (SQA), and describe the design requirement of SQA systems. Finally, we propose the generalized DeepSQA framework.

3.1 Sensory Question Answering (SQA)

Unlike visual, acoustic, or textual data, sensory data collected from sensors, e.g., IMUs and barometers, naturally cannot be understood by humans. All humans are "sensory impaired" to the sensory data due to the non-capability of proper sensory abstraction and the deficiency of a standard vocabulary describing different phenomena and their characteristics. Sensing using state-of-the-art deep learning models proves to be an effective way to comprehend the opaque sensory data. However, outputs of deep learning models are restricted to a pre-defined set of labels, limiting the inferencing flexibility significantly. Also, in order to make various inferences, users need to train different models, which requires large amounts of labeled data and compute power. Based on these limitations, it is essential to have an intelligent system to help humans understand sensory data in the form of question answering. We envision a system that gathers all sensory data from distributed heterogeneous sources. When the user asks arbitrary natural language questions, the system processes the received data, reasons about the spatial-temporal relationships between events, and provides correct answers in the form of natural language to humans.

Here we formally define the Sensory Question Answering (SQA) problem as follows. Consider a sensor network where data d_i of different modalities are continuously collected by a set of n distributed sensors. At some time point t, the user asks a question q in natural language format to the system, which is expected to output the correct answer a based on the context data D. $D = \{d_1, d_2, ..., d_n\}$, where $d_i = \{d_i^{t-k+1}, ..., d_i^{t-1}, d_i^t\}$ is a sequence of data collected by the i-th sensor. The number k represents the maximum length of history considered when answering the question, and it is usually determined based on different applications and system memory limitation. So this SQA system learns to model the conditional probability of $p(a|q, D)$.

Based on the formulation above, the SQA system needs to satisfy a set of requirements. Firstly, it should have the ability to process and fuse raw, multimodal sensory data collected directly from heterogeneous sensors. Secondly, the SQA system should be capable of analyzing natural language questions to understand what the various tasks are during the inference time. Thirdly, complex temporal and spatial dependencies between different sensory modalities, and more importantly, the correlation between questions and sensory contexts, need to be explored and captured by the SQA system so as to answer the question correctly.

3.2 SQA Architecture Design

As shown in Figure 2, we propose a generalized framework called DeepSQA. Here, we adopt the idea of question answering in other domains like VQA to our SQA problem. Basically, questions and images are first processed by two different paths to get compact representations, and then an SQA module is applied to analyze these representations and predict answers.

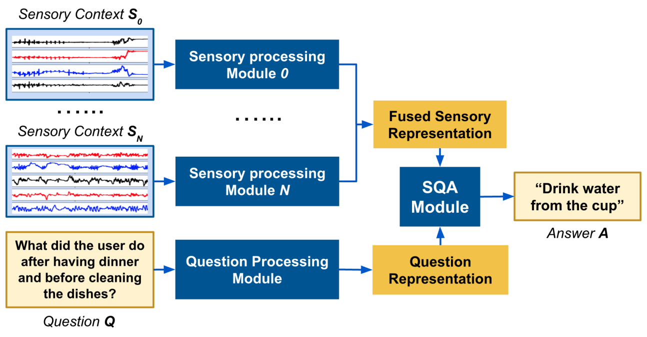
Sensory Context S_0
Sensory processing Module 0
......
......
Sensory Context S_N
Sensory processing Module N
Fused Sensory Representation
SQA Module
"Drink water from the cup"
Answer A
What did the user do after having dinner and before cleaning the dishes?
Question Processing Module
Question Representation
Question Q

Figure 2: The Generalized DeepSQA Framework

Sensory and Question Representations The question q with M words in natural language form is first embedded into a sequence of M embedding vectors using an embedding matrix, which is either pre-trained on a large text corpus, or learned together with other parameters during the training time. The embedding sequence is then processed by a recurrent-based structure, such as a multi-layer LSTM or bidirectional-LSTM network, to get the question representation.

In order to calculate the sensory representation, sensory processing modules are required to extract informative features from multimodal raw sensory data.

In the visual domain, a variety of pre-trained models are off-the-shelf. These models are trained on population-scale data of immense size, like the ImageNe[8] and Microsoft COCO[25] datasets. They are capable of processing and extracting informative visual features that can be used to perform a bunch of different downstream tasks, e.g., image classification, object detection, semantic segmentation, etc. As a result, in the task of visual Q & A, these pre-trained models can be used to obtain visual representations at both image-level and object-level directly.

However, for sensory question answering, such a general-purpose model pre-trained on population-scale data is not available, given the absence of a enough large-scale sensory dataset in both the sensing and machine learning communities. Because of the heterogeneity, data from different sensors have different modalities and sampling rates, and data collected in different locations, on different users, might vary significantly. As well-studied in prior work [7], a machine learning model trained on one sensory dataset would not perform well with other sensory data. This makes it impossible for us to use pre-trained models to get sensory representations.

Consequently, in DeepSQA, we use data-specific sensory processing modules instead. The sensory processing modules are designed based on the size and modality of raw sensor data. A popular structure is the Convolutional LSTM network (ConvLSTM), which can be used for extracting relevant sensory features and reasoning about their long-term temporal dependencies. An effective sensory processing module is crucial in the DeepSQA framework, as the downstream question answering task requires accurate context information to make correct predictions.

We also investigate using an auto-encoder-based approach to extract sensory features. The auto-encoders can be trained in a self-supervised manner with no annotation required. The intermediate bottleneck layer, which squeezes the original high-dimensional time-series sensory data into low-dimensional vectors, can be used as the sensory representations. However, because of the lack of training guidance, the sensory representations generated by the auto-encoders are not informative enough and can lead to huge performance sacrifice. Therefore, in this paper, we omit this method, and focus on the ConvLSTM based sensory processing module.

Reasoning with SQA Module In the DeepSQA, an SQA module is designed to find the inter-correlation between sensory and question data, perform reasoning, and predict final answers to complex compositional questions. Instead of explicitly decomposing the reasoning into multiple semantic sub-tasks as humans do, we adopt a neural-network-based approach to perform complex reasoning implicitly. This choice is motivated by QA research in other domains, which showed that neural network models perform better when they use deep, extracted features rather than using human-engineered features. Different techniques can be used in this SQA module, such as the simple convolutional RNN with multimodal data fusion, bi-linear pooling, and the attention mechanism. We will detail different models based on DeepSQA framework in Section 5.

Although there are some other techniques [2] that show superior performance than the purely neural approach in the VQA domain, they either require object-level or semantic features as input, or need additional supervision to decompose questions into semantic sub-tasks. These methods are not suitable in our case of SQA, simply because sensory data are not human-understandable, and the usage of semantic knowledge is not an option.

All the different modules in DeepSQA are stacked and connected together, and the entire system can be trained in an end-to-end fashion.

To train and evaluate different SQA systems along with a set of baselines, a large and diverse SQA dataset is required. However, because of the opacity of sensory data, humans cannot provide answers to question based on sensory contexts directly, and hence it is impossible to create an SQA dataset using the crowd-sourcing method as the VQA does. In this work, we propose a method SQA-Gen, which can generate SQA data based on a labeled source sensory dataset. Using this tool, we generate the OppQA dataset, that questions human activities and their temporal relationships. In the next section, we detail our tool for transforming a sensory dataset into an SQA dataset.

4 SQA-GEN: SQA DATASET GENERATION TOOL

In this work, we introduce a tool SQA-Gen for creating SQA data from a source sensory dataset. Using SQA-Gen, we generate the OppQA, a human activity sensory question answering dataset, based on the OPPORTUNITY [29] data. In this section, we first describe the data generation method we use in SQA-Gen, and then show the statistics about our generated OppQA dataset. We also need to mention that this generalized tool SQA-Gen can also be used to create new SQA datasets with other labeled source sensor data. We make the OppQA dataset and SQA data generation tool SQA-Gen available online to facilitate future research in the SQA field.

The context of the SQA dataset needs to have the characteristics of regular sensory data, showing the necessity of being multimodal, and involving multiple users with multiple devices. Besides, the sensory context needs to be real time-series data collected by sensors. Synthesized context data are unrealistic [1, 18], and the SQA models trained on that would not have satisfactory performance when deployed in real scenarios. Because of sensory data's opacity to humans, it is difficult to ask a human generator to create questions and answers based on sensory context manually. Therefore, in this work, we are using an automatic approach to generate sensory questions and answers.

Although the automatic generation may not provide many linguistic variations on natural language questions and answers, it has several other benefits. Firstly, it always provides objective and correct answers to even complex and compositional questions. In comparison, humans sometimes fail to give right answers [18], especially to complicated questions. Secondly, automatic generation is a scalable approach that can be applied to different source datasets at a minimal cost. In Section 6, we use SQA-Gen to generate multiple variants of OppQA with different parameter configurations, and also apply SQA-Gen to a new sensory dataset, ExtraSensory [32], to obtain new SQA data in a different domain. The scalability of SQA-Gen helps evaluate and train SQA models efficiently. It allows us to investigate SQA models' performance under different settings and also provides abundant amounts of data for model training. Thirdly, it is evident that the automatic QA generation shows higher efficiency with lower costs than human creation.

4.1 Source Data Selection

In this work, we choose the domain of human activity analysis to evaluate the SQA systems, and select the OPPORTUNITY Dataset[29] as our source sensory dataset. The OPPORTUNITY is a dataset used for benchmarking human activity recognition (HAR) tasks. The data are collected with wearable, object, and ambient sensors. Here we are only using the seven body-worn inertial measurement units (IMU) sensors to make inference on user activities. These sensors are located at different parts of the body: Left lower arm (LLA), Left upper arm (LUA), Right lower arm (RLA), Right Upper arm (RUA), Back of the torso(BACK), and Left/Right shoes(L-SHOE/R-SHOE). These inertial measurement units provide readings of: 3D acceleration, 3D rate of turn, 3D magnetic field, and orientation of the sensor with respect to a world coordinate system in quaternions. The IMU data are collected at a fixed sampling rate of 30Hz.

OPPORTUNITY also provides a rich set of annotations, including three different hierarchies: high-level activities, modes of locomotion, and low-level activities. Here we are using the two hierarchy of labels: locomotion activities(sit, stand, walk, lie, other) and low-level activities, including 17 different micro activities such as opening and closing doors, shelves, drawers, drinking tea, etc. This enables us to ask questions that reason about the interactions between different levels of activities, for example, "Did the user drink tea while he is standing?". OPPORTUNITY collects data on four distinct subjects. For each subject, six separate runs were recorded. The total length of the data is about 8 hours.

4.2 SQA Data Generation

With the labeled source sensory data, we are able to generate an SQA dataset. Inside SQA-Gen, the sensory context, the question, and the answer are all accompanied with a semantic representation to enable automatic machine generation. The entire data generation pipeline is illustrated in Figure 3, which we describe in detail below.

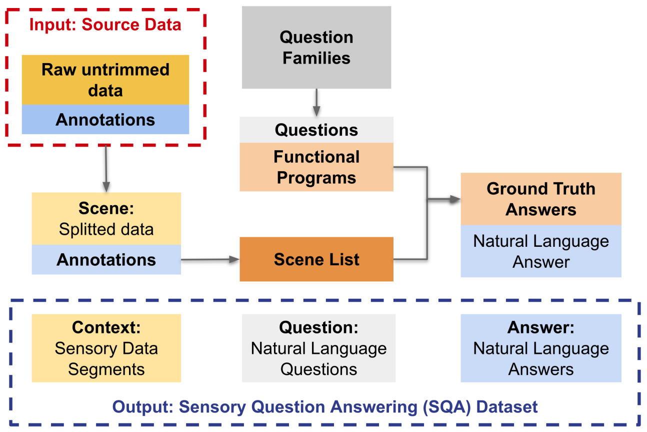

Figure 3: Data generation Pipeline

Sensory Context Generation. The untrimmed source OPPORTUNITY data are first split into data sequences with appropriate length, which are then used as the sensory scene (context) for the SQA task. The context length is an adjustable parameter of SQA-Gen, which determines the maximum length of history considered when answering the question in real-time. In OppQA, we choose the context length to be one minute, which means that the past 1800 data samples are used to answer the question. The data splitting is performed using a sliding window approach on the original time-series data. The length of the sliding window is one minute (1800 timesteps), and we use a stride of 20 seconds (600 timesteps) to avoid having two consecutive sensory contexts with too much information in common.

Scene Representation. For each sensory scene, we have its associate label sequence obtained from OPPORTUNITY's two hierarchies of annotations. The label sequence provides information about what the user is doing at every timestep. Corresponding to the two-hierarchy annotations, we create two scene lists using the label sequences as the semantic representation for each sensory context. A scene list $\mathbf{A} = \{A_1, A_2, ..., A_n\}$ contains n activities happening in a sequential order. For each activity $A_i = \{y_i, d_i, s_i\}$, the activity type y_i, duration d_i and starting time s_i is stored.

To create the scene lists, we traverse the label sequences and aggregate the consecutive data samples with the same label to a single activity. The "other" activity is omitted here. With the semantic representation of the sensory scene, we can easily reason about the relationships between different activities in an automatic manner.

We formally define the temporal relationships between different activities. We say activity A_i is after/before A_j if $i > j$ or $i < j$, and A_i is right after/before A_j if $i = j + 1$ or $i = j - 1$. And for the "While" relationship: suppose A and B are two scene list of different hierarchies, then the user did A_i while doing B_j only when the starting and ending point of A_i is within the range of $[B_j.s, B_j.s + B_j.d]$. We assume that the activities within the same hierarchy are mutually exclusive, so it is impossible to have users performing different activities of the same hierarchy at the same time.

Question Generation. Inspired by [14, 18], SQA questions are generated automatically using a template-based method. In the SQA-Gen tool, we create a set of 16 different question families, covering different question types: Action Query, Time Query, Existence, Counting, Action Comparison, Value Operation, etc. For each question family, a functional program template is used for constructing functional programs. With this program, several text templates exist for creating questions in different natural language forms. We need to mention that new question families and text templates can be easily added to SQA-Gen to generate new questions based on diverse requirements.

A functional program is composed of a set of functional building blocks. Similar to [18], we have a function catalog that deals with different operations for SQA reasoning. These functions can be combined in different ways with different input parameters to create an infinite number of questions with arbitrary complexity.

For example, the question "What did the user do before opening the fridge and after closing the drawer?" is generated using the text template "What did the user do [Relation] [Activity] [Combinator] [Relation] [Activity]", while filling in "before", "after" as relations, "and" as the Combinator, "open the fridge", "close the drawer" as activities. The functional program "query_action_type(AND(relate(before, open the fridge), relate(after, close the drawer)))" is instantiated using program template "query_action_type(Combination(relate(Relation, Activity), relate(Relation, Activity)))".

To generate questions using text templates, we need to take care of the tense, person, and plurality to avoid grammar mistakes. We take these three elements as additional parameters of the activity in the text templates. After generating a valid question, we also apply a synonym change in order to increase language diversity.

The functional program can be directly applied to the scene lists of the sensory scene to get the correct answer, which is then translated into natural language. At this stage, the sensory context, the question, and the answer are all generated.

Question Type and Answer Balancing. We slide the time window across the untrimmed time-series sensory data to get various sensory scenes. For each sensory scene, we construct its semantic representation *scene lists*, and based on the activities involved in this scene, we apply all possible functional programs to the scene lists to get the correct answers. Sometimes the questions are ill-posed, and the answers could not be obtained. For example, the question "What did the user do after closing the door?" would be *ill-posed* if the user closed the door more than once in the time window, and we could not identify which "close the door" activity the question is referring to. These ill-posed questions are rejected.

After traversing all the time-series data and iterating all the possible functional programs, we generate an initial SQA dataset, which includes around 35 million question & answer pairs and 1362 sensory contexts. However, the current SQA dataset has an unbalanced question type distribution and biased answer distributions. This would lead to inefficient training and possibly performance degradation. Therefore, we downsample the questions based on their types to control the question data type composition.[2]

More importantly, we balance the global answer distribution of the dataset, to avoid question-conditional biases, which allow learners to make educated guesses without understanding the sensory contexts. To do this, we first calculate the global answer distribution for each question type, and then downsample the questions with the most frequent answer to a pre-defined ratio of 80%. We repeat this operation iteratively, until the stopping criteria are met: the question numbers, or the ratio of the most frequent answer is below certain pre-set thresholds.

After question resampling and answer balancing, the SQA dataset has more balanced answer distributions and question type composition. Details are illustrated in Figure 4.

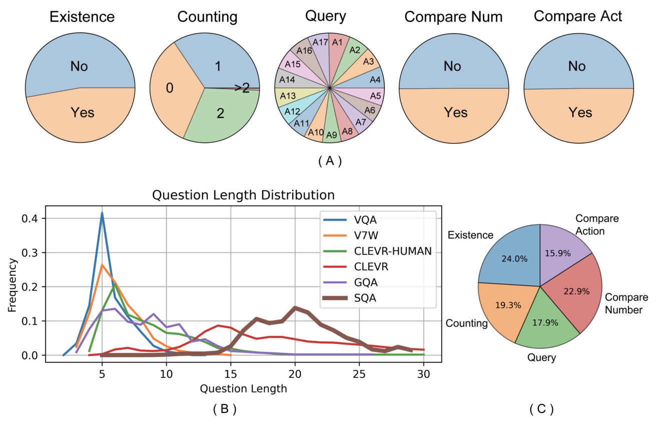

Figure 4: Statistics of OppQA dataset. (A):Global answer distribution of different question types. (B): Question length distribution compared with other VQA datasets. (C): Question type composition. Question type "Time Query" with non-categorical answers is excluded in this figure.

4.3 Summary of Generated OppQA Dataset.

To create the OppQA dataset, we use a total of 16 question families and more than 110 text templates. Table 1 shows examples of generated questions for different types, and also their possible answers.

Table 2 shows the statistics of OppQA dataset. It includes 1362 unique sensory scenes, and a total of 91 thousand questions, 72 thousand of which are unique. Among them, 39 thousand unique queries are used. The question query represents the functional program used to generate the question. Diverse questions with the

[2]We exclude the questions of "Time Query" type in OppQA dataset, since these questions have non-categorical answers. SQA models with regression tasks are left to future work.

Q Type	Question Example	Possible Answer
Existence	Is it true that the user closed the door after opening the door?	Yes, No
Counting	How many times did the user close the door?	<integer >
Action Query	What did the subject do after cleaning the table?	open the door, wash dishes, drink water, ...
Time Query	How long did the user wash dishes?	<float >
Action Compare	Confirm if the user performed the same action proceeding and following opening the fridge?	Yes, No
Number Compare	The user toggled the switch for the same times before and after drinking water?	Yes, No

Table 1: Question Examples and Possible Answers in OppQA.

same semantic meaning share the same query. As shown in Figure 4 (B), OppQA has an average question length of 18 words, and is complicated enough compared with other popular VQA datasets. It can be used as a benchmark for evaluating SQA models.

The OppQA data is split into a training set and a testing set. The training set contains SQA data generated on the first two Activity-of-Daily-Living (ADL) runs and a drill run of users 1-4, and the rest of the runs are used to generate testing data. The training data and testing data do not share sensory context, but they have overlapping question queries. Table 2 lists the details of OppQA.

Split	Sensory Contexts	Questions	Unique Questions	Unique Queries
Total	1,362	91,412	72,936	38,922
Training	730	74,470	62,789	35,262
Testing	632	16,942	14,643	6,275

Table 2: Statistics for the *OppQA* dataset.

5 DEEPSQA MODELS AND IMPLEMENTATIONS

In this section, we describe the structures and implementation details of the proposed SQA models, along with a set of baselines.

All of the models that process context sensory data first use a convolutional-LSTM network to get the sensory representations. The ConvLSTM network is composed of two convolutional modules(which contain a convolution layer with 1×3 kernel, ReLU activation and a maxpooling layer), followed by an LSTM layer and a fully-connected layer to generate a 128-dimension dense embedding vector. Both the LSTM layer and fully-connected layer have 128 hidden units. Instead of processing sensory data from different sources using different neural networks, we perform an early fusion on 77 channels of sensory reading from all the seven distributed sensors, which leads to better performance on this dataset. Therefore, the size of input sensory data is 77×1800, with 1800 specifying the window length of a one-minute window. This parameter is also changed in later simulations when we evaluate the SQA performance with respect to the task complexity.

All the models, that process questions, use LSTM-based models to get question representations. Every word in the question is first embedded into a 300-dimensional representation using a pre-trained GloVe[26] word embedding matrix, unless otherwise noted. The GloVe word vectors are pre-trained with six billion tokens with a vocabulary of 400K words. We use in total two different LSTM-based structures to extract question representations. Details will be discussed below.

In the evaluation, we exclude the questions asking about the duration of activities, which give various non-integer numbers as answers. We then formulate the SQA task as a multi-class classification problem[3], using the top 27 answers as the classification labels. Therefore, the output of every model should be a distribution of predicted scores of these candidate answers.

5.1 Baseline Models

We adopt a representative subset of methods from VQA: baselines that predict answers based on statistics of training data (Prior and Prior-Q), baselines using only question data (LSTM), or using only sensory data as input (ConvLSTM). Since sensory data are opaque to humans, we cannot use the crowd-sourced method to collect human answers and evaluate human performance on this task. As an alternative, we use a Neural-Symbolic approach with pre-trained native sensory classifiers, and perfect question logic knowledge to mimic human performance. These methods are described in details as follow:

- **Prior:** As [3], this baseline answers the most frequent answer in the training dataset for all questions, which is 'No' in OppQA dataset.
- **Prior-Q:** Similar to the *Prior* method, this model predicts answers based on training data statistics. For each question type, it predicts the most frequent training-set answer.
- **LSTM:** Similar to the "LSTM Q" in [3], questions are first processed with learned word embeddings, and analyzed by a word-level two-layer LSTM model, where each layer contains 128 hidden units. The output question representation, which is the final state of LSTM, is then fed into an MLP network using two hidden layers with 128 units to predict the distribution over answers. This method uses no context information as it is "sensory-blind," so it can only model question-conditional bias.
- **ConvLSTM:** Inspired by [18] and [3], this "question-blind" model only uses sensory context. Sensory data are first processed using the ConvLSTM network, and then the answer probability distribution is predicted by an MLP with one hidden layer that has 128 units and a softmax output layer. This baseline model predicts answers by guessing the questions based on the statistics of QA training data.
- **Neural-Symbolic:** In this method, we employ two activity classification models recognizing user activities and locomotion using sensory data at every time step. We then use 100% correct, hard-coded logic rules to analyze the classification result and answer the question. These rules are the same as those used for constructing the SQA dataset. Basically, they are functional modules in Python, hosting all logic operations, and are selected based on different questions. The activity classification models are ConvLSTM structures that contain two convolutional layers, followed by an LSTM layer

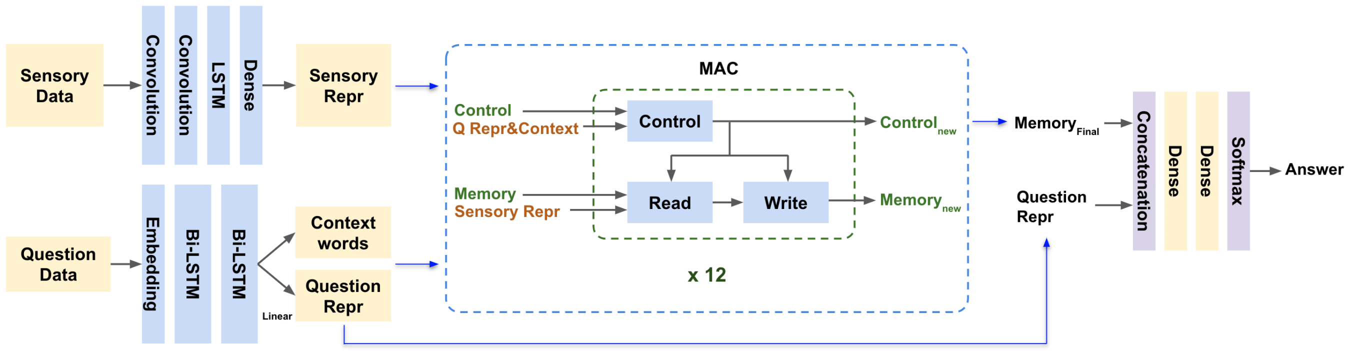

Figure 5: Overview of DeepSQA-CA Model

and a fully-connected layer to map the sensory data to the predicted activity. The convolutional kernel size is 1×3, and two convolutional layers are followed by batch normalization layers[15], which address the internal covariate shift problem of neural networks. Sixty-four feature maps are used in each convolutional layer, and 32 hidden nodes in the LSTM layer. The activity classification models are trained natively on the original OPPORTUNITY dataset, with an accuracy of 80.13% and 74.13% on the activity classification and locomotion classification tasks, respectively. These numbers could represent the performance of state-of-the-art sensory models on this dataset.

5.2 DeepSQA-based models

Based on the DeepSQA framework, we propose and evaluate three different models that process both sensory contexts and questions together, analyze and reason about the spatial-temporal relationships, and draw the final answers. These models are chosen as representatives, as they employ the state-of-the-art mechanisms and structures in the multimodal deep learning and the question answering domains. The model DeepSQA-ConvLSTM, uses a simple but effective elementwise multiplication to fuse the question and sensory representations together and predict answer based on it. The model DeepSQA-SA, learns how to search for the features in sensory contexts that are related to the answer using attention weights calculated based on the questions. The model DeepSQA-CA, decomposes the questions into multiple explicit reasoning steps, customizes the weighting of the context feature vector for each step, and performs iterative reasoning processes to get the answer. Details of these models are discussed as follows.

DeepSQA-ConvLSTM: This model combines the ability of both sensory processing and question reasoning in the LSTM and ConvLSTM baselines. Since the representations of both modalities have the same dimension of 128, the model first fuses the sensory and question representations using an element-wise multiplication, and then feeds the combined feature to a two-layer MLP (with 128 hidden nodes) to get the prediction of the answer distribution.

DeepSQA-SA: In this model, we use the Stack Attention(SA) [22, 37] mechanism to fuse the sensory and question information, and then predict answers based on new attended features.

Specifically, the questions and sensory contexts are first processed using an LSTM and a ConvLSTM models to get compact representations in the latent space. These two representations are then concatenated together as a combined feature. In the Stack Attention network, we use a two-layer CNN with 1×1 kernels to calculate the spatial attention weights. The first and second convolutional layers' activation functions are ReLU and Softmax, to calculate the normalized probabilities over all the spatial locations.

The calculated attention weights are then multiplied with the sensory representation to get the attended sensory feature, where the information relevant to the question is highlighted.

We use a glimpse number of two to get two sets of different attended sensory features. These features are concatenated with the question representation again and fed to a 2-layer MLP network with 1024 hidden nodes. The final answer is predicted by the last softmax layer.

DeepSQA-CA: This method uses the Compositional Attention (CA) [13] mechanism to perform multi-step reasoning over the sensory and question data, by employing a recurrent structure which strings together p MAC cells, each responsible for performing one reasoning step. A MAC cell is the basic module in this recurrent architecture.

In this model, the sensory contexts are processed using the same ConvLSTM network, while the question data are processed differently. The original question with word tokens is first converted into a sequence of word embeddings. Instead of using pre-trained GloVe word embeddings, we learn the embedding matrix with other network parameters during the training time. The embedding sequence is then processed by a bidirectional LSTM network, which trains two instead of one LSTMs on the input sequence in two opposite directions. The question representation is then the final hidden states' concatenation from the forward and backward LSTM passes. In addition to the question representation, we also generate a contextual word sequence, by storing the sequence of output states for each word in the biLSTM network.

After getting the sensory and question representations, we feed them to the recurrent MAC structure to perform multi-step reasoning. Each MAC cell is composed of a control, read, and write unit, which operate over control and memory hidden states. The control unit attends to different parts of the question, to update the control state that represents the reasoning operation at each

	Baselines					DeepSQA		
	Prior	PriorQ	Neural Symbolic	ConvLSTM	LSTM	SA	ConvLSTM	CA
Overall	41.57%	54.82%	42.75%	44.92%	65.04%	59.74%	67.63%	**72.38%**
Binary	53.27%	65.26%	51.10%	57.56%	68.33%	61.78%	71.81%	**76.51%**
Open	0.00%	17.74%	13.09%	0.00%	53.35%	52.49%	52.78%	**57.67%**
Existence	66.63%	66.34%	46.15%	39.97%	66.99%	67.20%	69.76%	**72.69%**
Counting	0.00%	35.99%	30.86%	0.00%	60.71%	58.94%	59.06%	**63.31%**
Action Query	0.00%	4.29%	0.00%	0.00%	47.92%	47.74%	48.16%	**53.52%**
Num Comparison	53.59%	72.45%	66.61%	63.21%	70.73%	63.57%	71.91%	**76.61%**
Act Comparison	37.99%	37.99%	0.00%	55.64%	61.02%	49.57%	73.62%	**80.21%**

Table 3: Overall results of models trained and tested on OppQA dataset

time step. The read unit extracts relevant information out of the sensory context with the guidance of the control state. The write unit integrates extracted information into a memory state, which becomes the input of the next MAC cell.

The input of a MAC cell are the previous control and memory states, together with the sensory context representation, question representation, and contextual words. The initial control and memory states are initialized learned parameters.

At last, the final memory and question representation are concatenated and fed to a 2-layer MLP with a softmax classifier to predict the distribution over candidate answers. An overview of this model is demonstrated in Figure 5.

In our implementation, we set the reasoning step value p as 12, which means that the recurrent structure has 12 MAC cells. The dimension of hidden states (control and memory) is 512, and the final MLP network has two layers with 1536 and 512 hidden nodes.

6 EVALUATION

In this section, we empirically evaluate the proposed DeepSQA framework by comparing the model performance with baseline methods. We use the created OppQA dataset and its variants to test the effectiveness and robustness of SQA models on reasoning about the spatial-temporal dependencies of human activities.

6.1 Implementation Details

In our experiment, the DeepSQA-CA model is implemented in Pytorch, and the other models are implemented with TensorFlow and Keras frameworks. Although different frameworks are used, the learning settings have been made consistent for a fair comparison. All the models are trained and tested on a desktop machine with two Nvidia RTX Titan GPUs. During the training, a 0.15 dropout rate is applied to the convolutional, dense, and LSTM-based layers for the ConvLSTM and MLP models, and a $1e-4$ weight decay is applied to the DeepSQA-CA model to avoid overfitting. We train all the models for 40 epochs using Adam optimizer with a learning rate equal to $1e-4$ and a batch size of 64.

6.2 General Observations

We first evaluate the DeepSQA models and baselines on the standard OppQA dataset, which covers questions of 15 types, and reasons about human activities over a time window of one minute (1800 samples). The maximum question length is 31 words, and the number of unique candidate answer is 27, with non-integer answers excluded. The overall performance of each method is shown in Table 3. We break down the performance based on different question types: existence, counting, action query, number comparison, and action comparison. Among them, counting and query are further classified into open-ended questions and the remaining into binary questions.

The *Prior* method in the first column predicts "No" for all the questions, and gets a 0% accuracy on open-ended questions and 53.27% on binary questions, which is a little bit higher than 50% of a random guess. This indicates that the answer distribution in the OppQA dataset is balanced with minimum global bias. Taking question types into consideration, the *PriorQ* method gets better performance, but it still could not predict answers to open-ended questions well. On the "Action Query" question type, *PriorQ* can only get an accuracy of 4.29%, proving that it is indeed a challenging task to predict the correct answer out of 27 candidate answers.

Albeit using pre-trained activity classification networks and perfect reasoning logic, the *Neural Symbolic* method shows bad performance on OppQA. This is because the logical rules do not anticipate the inaccurate primitive events, as humans are bad at generating logic rules that work with noisy or even erroneous data. On the other hand, the deep learning models cannot make accurate inferences about the primitive activities for the Opportunity dataset (with an accuracy of around 70-80%). The errors in the noisy inferences accumulate and then dramatically reduce the answer correctness. Conversely, the end-to-end neural network approaches do not suffer from this problem. One possible explanation is that the neural network could learn to use reliable features, and compensate for the possible error to make the right prediction.

Generally, the models with DeepSQA framework show better performance than the baselines, especially the *DeepSQA-CA* model, which uses the compositional attention mechanism, gets the best performance on all of the question types, with an overall accuracy of 72.38%. The *DeepSQA-SA* model based on stacked attention shows inferior performance than some of the baselines. This proves that, although demonstrating good performance on visual QA tasks, the stacked attention does not work well on reasoning about temporal dependencies between human activities.

It's worth noting that the LSTM model can achieve similar performance to some of the DeepSQA models with over 60% overall accuracy *without* using any sensory context data. This is due to the fact that for some particular questions, the local bias still exists. For example, when the question is "What did the user do after opening the door?" the answer would be "Close the door" with a 90% probability. We also notice that this is a common case of many existing QA work in other domains[3]. Since we are using real data, it is inevitable to contain bias induced by predictable human behavior patterns.

Performance on Prime Dataset. Because of the unavoidable local bias in the dataset, we construct a "Prime Testing Set" based on the original testing data. In this *prime dataset*, we abandon the questions with only a single answer, and all the questions have more than one answer. This makes it impossible for the models to predict answers using only question data and without sensory context. This prime set is actually more challenging than the original testing set. In our experiment, among all SQA data samples, about half of them are prime data. In Table 4, we list the performance comparison of all the deep-learning-based models on the testing dataset and prime dataset. Generally, the accuracies on the prime dataset are lower than those on the testing set. DeepSQA-CA has the minimum performance degradation while maintaining good accuracy. Its accuracy loss is less than 2%. The baseline *ConvLSTM* model observes a performance improvement on the prime dataset. It is because that the non-prime questions with a single answer are answered incorrectly by this model, and removing those questions could passively increase the accuracy.

	Baselines		DeepSQA		
	ConLSTM	LSTM	SA	ConvLSTM	CA
Testing	44.92%	65.04%	59.74%	67.63%	**72.38%**
Prime	56.56%	60.39%	53.46%	64.90%	**70.48%**

Table 4: Overall performance on prime testing set

Robustness Against Linguistic Variations. The key motivation of proposing the sensory question answering task is to improve the inferencing flexibility of sensing systems, in a way that users can make arbitrary inferences during the run time in the form of natural language. However, for questions with the same semantic meanings, the natural language representations might be dramatically different. In order to test the robustness against linguistic variations of SQA systems, we create a "Rephrasing Testing set." This dataset contains different rephrasings for all the questions in the original testing set, generated using our question generation tool. We evaluate the four Deep-Learning-based models which take natural language question as input on the rephrasing dataset, and list the corresponding accuracies in Table 5. Basically, the performance on the rephrasing set is similar to the testing accuracy, indicating that Deep-Learning-based SQA models are robust to question rephrasing. In addition, we define a consistency score to further measure the model robustness on every query:

$$Consistency = \frac{\sum_{i=1}^{m} \mathbb{1}(a_i = a_{major})}{m}$$

The m represents the number of rephrasings for each question query, and a_{major} is the majority answer predicted by the SQA model. Table 5 shows that the consistency scores for all the models are greater than 95%, which proves their robustness to linguistic variations.

	LSTM	DeepSQA SA	DeepSQA ConvLSTM	DeepSQA CA
Testing Acc	65.03%	59.73%	67.63%	**72.38%**
Rephrasing Acc	65.86%	60.51%	68.56%	**72.86%**
Consistency	99.13%	98.97%	99.06%	96.28%

Table 5: SQA robustness to linguistic variations

6.3 Analysis by Question Complexity

To access the SQA model performance with respect to question complexity, we analyze the evaluation result of deep learning models by the question length, as shown in Figure 6. The models that do not take the question as input are omitted here. Intuitively, longer questions should contain more query information, and would be more challenging to answer than shorter ones. Surprisingly, we do not see a clear correlation between question length and SQA performance. One possible reason is that the number of words in questions cannot effectively reflect the questions' complexity. Some questions are more verbose than the others, and contain redundant or even useless information, while representing simple queries. This would lead to the result that SQA models have better accuracies on some longer questions than those on shorter questions.

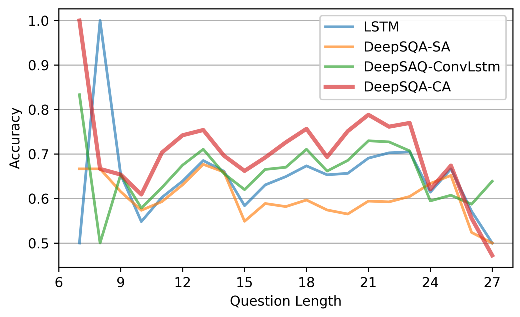

Figure 6: Models performance w.r.t. question length.

Instead of using the number of words as a question complexity measure, we use the number of required query operations to answer the questions to better capture the complexity. For example, questions like "What did the user do after washing dishes?" can be answered using three operations: 1. detect and localize the "washing dishes" activity; 2.Filter the activities after the "washing dishes"; 3. Query the type of filtered activity.

Since the binary and open-ended questions have different complexity, we first categorize the questions into two groups, and analyze them separately. As shown in Figure 7 (A), for binary questions, there is a linear decrease in SQA model performance with the number of operations increasing from one to three. However, the accuracies of questions requiring seven query operations are pretty high. It is probably because the number of this type of question is larger than others in the training dataset. Models trained on this training dataset learn to perform well on this type of questions.

Figure 7 (B) shows the SQA model performance changes on open-ended questions. Specifically, the accuracy of the DeepSQA-CA model decreases with the required query operations increasing. However, the other models show poor performance on questions

	Test-Familiar			Test-Novel		
	Binary	**Open**	**Overall**	**Binary**	**Open**	**Overall**
Prior	46.90%	0.00%	36.64%	46.55%	0.00%	36.28%
PriorQ	70.11%	18.51%	58.82%	68.86%	19.02%	57.87%
Neural Symbolic	51.12%	12.69%	42.71%	51.08%	13.51%	42.80%
CNN	58.09%	0.00%	45.38%	58.01%	0.00%	45.22%
LSTM	67.80%	53.54%	64.68%	66.91%	46.86%	62.49%
DeepSQA(san)	69.89%	53.38%	66.28%	67.72%	49.01%	63.60%
DeepSQA(convlstm)	71.96%	53.49%	67.92%	71.71%	48.90%	66.68%
DeepSQA(mac)	**73.03%**	**58.42%**	**69.83%**	**72.70%**	**54.30%**	**68.64%**

Table 6: Model performance generalization to novel questions

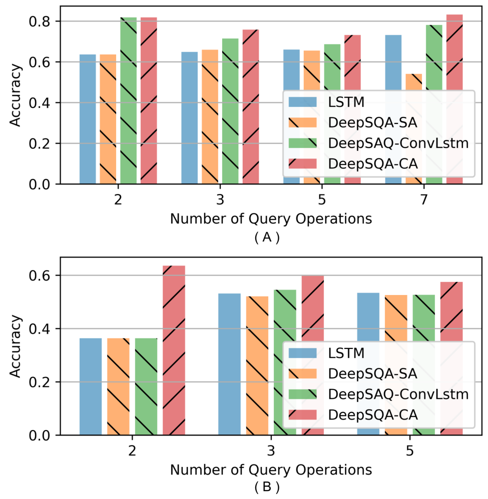

Figure 7: Models performance w.r.t. question complexity: (A) Binary and (B) Open-ended.

requiring two operations. We notice that the SQA models are sensitive to the amount of training data. A larger training dataset would generally lead to a better performance. Apart from these, we can find that the DeepSQA-CA model shows the best performance consistently regardless of the complexity of input questions.

6.4 Analysis by Context Complexity

We also perform a set of experiments using new datasets generated in the same way as the original OppQA, but with different context window lengths. Here, we generate four variant datasets, using a window length of 500, 750, 1000, and 1500 time steps.

Apparently, the SQA task with shorter context length would be simpler, since fewer activities are involved in the scene, and reasoning about them is easier. In Figure 8, we plot the bar chart of performance for all SQA models that take the sensory context as input. When the sensor context window is only 500-time-step long, the models like DeepSQA-CA and DeepSQA-ConvLSTM could get excellent performance with over 90% accuracies. The performance of DeepSQA-ConvLSTM is even slightly better than DeepSQA-CA. It is because within a short period, the complex SQA reasoning module is not necessary. With the length of sensory context window increasing, the accuracy of SQA models keeps decreasing, which means that the length of sensory context is a crucial factor affecting the SQA task's complexity. Generally, the DeepSQA-CA model shows the best performance consistently, especially when the sensory context is long and the task is challenging. This illustrates the effectiveness of the compositional attention mechanism in processing complex spatial-temporal dependencies on human activity reasoning.

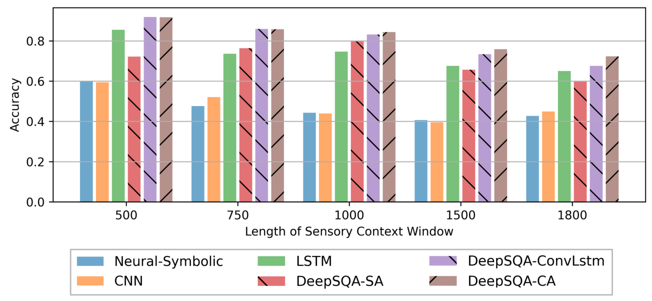

Figure 8: Model performance w.r.t. SQA task complexity.

6.5 Generalization to new questions

Another interesting characteristic of sensory question answering is its data-efficient training and generalization ability to new data. During the training, instead of learning to answer each specific question, SQA models learn to perform basic logical operations and predict answers based on operation results. At the inference time, SQA models are able to answer novel questions that they have never seen in the training dataset, but are composed of familiar activities and logic operations.

In this evaluation, we construct a new dataset *OppQA-generalize* to test the generalization ability of different SQA models. Specifically, this dataset adapts the same configuration as the original OppQA: with 77-channel data from 7 distributed sensors. We set the context window length to 1800, and the stride to 600 during generation. We split the data based on sensory context and question query to construct the training and testing set. The training set covers 80% of the sensory contexts, and 50% of all the unique question queries. The testing set contains the rest 20% of the sensory contexts, and all the unique question queries. The testing set is further divided into *Test-familiar* and *Test-novel* sets. The *Test-familiar* has question queries overlapping with the training set, and *Test-novel* has queries different from the training set. The details of this dataset are described in Table 7.

Split	Sensory Contexts	Questions	Unique Queries
Train	726	38,121	17,587
Test-Familiar	625	8,715	3,420
Test-Novel	627	8,227	3,305

Table 7: Statistics for the *OppQA-generalize* dataset.

We train different SQA models on the training set, and then test the model performance on *Test-familiar* and *Test-novel* sets separately. The accuracies are listed in Table 6.

As shown in the table, most models have a performance degradation when facing the unseen question queries in the *Test-novel* dataset. The degradation is more evident for open-ended questions, where the accuracies drop by around $4-7\%$, compared with $0-2\%$ for binary questions. The DeepSQA-CA model continues to show the best performance on both *Test-familiar* and *Test-novel* testing sets, and it has the lowest performance degradation. This result demonstrates that the DeepSQA model has better generalization ability to new question queries.

6.6 Discussion

Generation of Natural Language Answer. In this work, we formulate the SQA problem as a classification problem, where the output classes are the top K answers appearing in the training dataset. The Deep-Learning-based SQA models predict the answer distribution on the K classes, and the class with the highest probability is selected as the predicted answer. Several limitations exist in this approach. Firstly, the regression-type questions, such as Query-Time, are not enabled by the SQA models. Secondly, although the SQA models support answering diverse questions of different types by enumerating the possible answers in the training set, the output is still constrained to a limited set. In future work, we will augment the DeepSQA with an answer generator, which takes processed information from the proceeding reasoning module and creates natural language answers.

Neural-Symbolic DeepSQA framework. At the current stage, we use an end-to-end neural network structure for modeling the SQA task. So the prediction made by the model is not explainable. A significant drawback of this purely data-driven approach is that it's data-intensive during training, and also memory-consuming for storing the learned dependencies. Based on the evaluation result, we can observe that model performance degrades when facing longer sequences (e.g., longer sensory time windows, complex question queries). In future work, we would like to introduce the idea of neural-symbolic system [36, 38] to the SQA task, to get a more compact symbolic representation, and robustness on complex scenarios.

Evaluation on Different Sensory Datasets. In addition to OppQA, we also generated another SQA dataset using ExtraSensory[32] as source data. This dataset contains multimodal sensory data of multi-day length collected from 60 various users. It covers more than 50 labels about human daily activities, like bicycling, computer work, and sleeping. However, most of the SQA models show less satisfied performance on it. The poor performance is due to the challenge of deep learning models extracting useful information from ExtraSensory data. The model trained on ExtraSensory natively could only get around $< 60\%$ accuracy on the activity classification task. Consequently, the sensory representation could not provide much information to the DeepSQA. Admittedly, the vibrancy of visual question answering research largely depends on the mature visual information processing ability. We believe that the proposed DeepSQA framework and SQA-Gen tool in this paper could help facilitate the SQA research.

Generalization to Complex & Distributed Scenarios. Thus far, we have only discussed the evaluation on OppQA, where multimodal data from different sensors are fused together such that a single sensory processing module is used in DeepSQA models. We believe that DeepSQA can be generalized to SQA scenarios with multimodal data dispersed far apart. The complex spatial-temporal relationship between sensory data needs to be explored and reasoned to get the correct answer. In future work, we will deploy and evaluate DeepSQA system in real complex sensor network scenarios, with more rising challenges like time synchronization, sensory fusion, and noisy data annotation.

7 CONCLUSION

In this work, we present DeepSQA, a generalized framework for sensory question answering. By taking both sensory data and natural language questions simultaneously, DeepSQA is able to identify the tasks specified by questions and perform reasoning on sensory data accordingly. It reduces the laborious work of training new deep learning models when new tasks are introduced, and improves the inferencing flexibility. To evaluate SQA models, we introduce SQA-Gen, which automatically generates SQA datasets using labeled source sensory data. Based on this tool, we propose the OppQA dataset for benchmarking SQA model performance. Our work is the first one to address the SQA problem, and we hope the open-source tools and datasets can be used for follow up research on SQA models. Results on OppQA prove the effectiveness, reliability, and robustness of DeepSQA. Future work will focus on evaluating DeepSQA in real-world scenarios with distributed and multimodal sensory data.

ACKNOWLEDGMENTS

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement # W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] Jerome Abdelnour, Giampiero Salvi, and Jean Rouat. 2018. CLEAR: A Dataset for Compositional Language and Elementary Acoustic Reasoning. *arXiv preprint arXiv:1811.10561* (2018).

[2] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 6077–6086.

[3] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision.* 2425–2433.

[4] Pouya Bashivan, Irina Rish, Mohammed Yeasin, and Noel Codella. 2015. Learning representations from EEG with deep recurrent-convolutional neural networks. *arXiv preprint arXiv:1511.06448* (2015).

[5] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing.* 1533–1544.

[6] Ella Browning, Mark Bolton, Ellie Owen, Akiko Shoji, Tim Guilford, and Robin Freeman. 2018. Predicting animal behaviour using deep learning: GPS data alone accurately predict diving in seabirds. *Methods in Ecology and Evolution* 9, 3 (2018), 681–692.

[7] Diane Cook, Kyle D Feuz, and Narayanan C Krishnan. 2013. Transfer learning for activity recognition: A survey. *Knowledge and information systems* 36, 3 (2013), 537–556.

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition.* Ieee, 248–255.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[10] Nils Y Hammerla, Shane Halloran, and Thomas Plötz. 2016. Deep, convolutional, and recurrent models for human activity recognition using wearables. *arXiv preprint arXiv:1604.08880* (2016).

[11] Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. TAPAS: Weakly Supervised Table Parsing via Pre-training. *arXiv preprint arXiv:2004.02349* (2020).

[12] Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision.* 804–813.

[13] Drew A Hudson and Christopher D Manning. 2018. Compositional attention networks for machine reasoning. *arXiv preprint arXiv:1803.03067* (2018).

[14] Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 6700–6709.

[15] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning.* PMLR, 448–456.

[16] Yunseok Jang, Yale Song, Youngjae Yu, Youngjin Kim, and Gunhee Kim. 2017. Tgif-qa: Toward spatio-temporal reasoning in visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2758–2766.

[17] Lu Jiang, Junwei Liang, Liangliang Cao, Yannis Kalantidis, Sachin Farfade, and Alexander Hauptmann. 2017. Memexqa: Visual memex question answering. *arXiv preprint arXiv:1708.01336* (2017).

[18] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2901–2910.

[19] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision.* 2989–2998.

[20] Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. 2018. DVQA: Understanding data visualizations via question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 5648–5656.

[21] Kushal Kafle, Robik Shrestha, Scott Cohen, Brian Price, and Christopher Kanan. 2020. Answering questions about data visualizations using efficient bimodal fusion. In *The IEEE Winter Conference on Applications of Computer Vision.* 1498–1507.

[22] Vahid Kazemi and Ali Elqursh. 2017. Show, ask, attend, and answer: A strong baseline for visual question answering. *arXiv preprint arXiv:1704.03162* (2017).

[23] Oscar D Lara and Miguel A Labrador. 2012. A survey on human activity recognition using wearable sensors. *IEEE communications surveys & tutorials* 15, 3 (2012), 1192–1209.

[24] Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L Berg. 2018. Tvqa: Localized, compositional video question answering. *arXiv preprint arXiv:1809.01696* (2018).

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision.* Springer, 740–755.

[26] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP).* 1532–1543.

[27] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2017. Film: Visual reasoning with a general conditioning layer. *arXiv preprint arXiv:1709.07871* (2017).

[28] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).

[29] Hesam Sagha, Sundara Tejaswi Digumarti, José del R Millán, Ricardo Chavarriaga, Alberto Calatroni, Daniel Roggen, and Gerhard Tröster. 2011. Benchmarking classification techniques using the Opportunity human activity dataset. In *2011 IEEE International Conference on Systems, Man, and Cybernetics.* IEEE, 36–40.

[30] Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Incel, Hans Scholten, and Paul JM Havinga. 2016. Complex human activity recognition using smartphone and wrist-worn motion sensors. *Sensors* 16, 4 (2016), 426.

[31] Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding stories in movies through question-answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 4631–4640.

[32] Yonatan Vaizman, Katherine Ellis, Gert Lanckriet, and Nadir Weibel. 2018. Extrasensory app: Data collection in-the-wild with rich user interface to self-report behavior. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems.* 1–12.

[33] Jindong Wang, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. 2019. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters* 119 (2019), 3–11.

[34] Wei Wang, Alex X Liu, Muhammad Shahzad, Kang Ling, and Sanglu Lu. 2017. Device-free human activity recognition using commercial WiFi devices. *IEEE Journal on Selected Areas in Communications* 35, 5 (2017), 1118–1131.

[35] Xiao-Wei Wang, Dan Nie, and Bao-Liang Lu. 2014. Emotional state classification from EEG data using machine learning approach. *Neurocomputing* 129 (2014), 94–106.

[36] Tianwei Xing, Luis Garcia, Marc Roig Vilamala, Federico Cerutti, Lance Kaplan, Alun Preece, and Mani Srivastava. 2020. Neuroplex: learning to detect complex events in sensor networks through knowledge injection. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems.* 489–502.

[37] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2016. Stacked attention networks for image question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 21–29.

[38] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. 2018. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *Advances in neural information processing systems.* 1031–1042.

[39] Yu Zheng, Like Liu, Longhao Wang, and Xing Xie. 2008. Learning transportation mode from raw gps data for geographic applications on the web. In *Proceedings of the 17th international conference on World Wide Web.* 247–256.