UNIVERSITY OF CALIFORNIA

Los Angeles

Robust and Interpretable Predictions for Multimodal Sensor Systems

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Electrical and Computer Engineering

by

Jeya Vikranth Jeyakumar

2022

ABSTRACT OF THE DISSERTATION

Robust and Interpretable Predictions for Multimodal Sensor Systems

by

Jeya Vikranth Jeyakumar
Doctor of Philosophy in Electrical and Computer Engineering
University of California, Los Angeles, 2022
Professor Mani B. Srivastava, Chair

Smart IoT devices, smartphones, and wearables are penetrating every aspect of our daily lives. These devices are equipped with various sensing modalities, including video, audio, inertial sensors, lidars, etc., that enable multiple sensing applications. Research has shown that rather than operating each sensor in isolation, combining information from multiple sensing streams boosts performance. This method is known as multimodal sensor fusion and Human Activity Recognition(HAR) is one of the applications that benefits from using multiple sensors. In recent years, deep learning algorithms have been shown to achieve high accuracies in HAR using multimodal sensor data. However, in order to design a reliable HAR system, the following challenges still need to be addressed. The first challenge is the heterogeneity of the sensing devices. This arises as the set of devices monitoring a person may vary over time or the devices might have different sampling frequencies. And the second challenge is deep neural networks (DNNs) are considered black boxes because studying their structure often provides little to no insight into the actual underlying mechanics. It is hard to look "into" the network and ascertain why the model selects specific features over others during training, thereby making the predictions from the DNNs not trustworthy to the end-users. This lack of trust prevents the adoption of DNN models in health-related applications and other high-stakes applications where sensitive decisions mandate a sufficient accompanying

explanation. Therefore, this dissertation proposes methods to generate accurate predictions robust to the heterogeneity of devices by making opportunistic use of information from available devices and providing human-understandable explanations accompanying each prediction to the end-users.

First, we propose a solution to address the challenges related to the heterogeneity in sensor devices for activity recognition in our work 'SenseHAR.'We design a scalable deep learning-based solution in which each device learns its own sensor fusion model that maps the raw sensor values to a shared low dimensional latent space which we call the 'SenseHAR'-a virtual activity sensor. The virtual sensor has the same format and behavior regardless of the subset of devices, sensors availability, sampling rate, or device location. *SenseHAR* helps machine learning engineers to develop their application-specific (e.g., from gesture recognition to activities of daily life) models in a hardware-agnostic manner based on this virtual activity sensor.

Next, we address the issue of explainability for activity recognition in deep learning models. We first identify the most preferred post-hoc explanation technique for classification tasks across different modalities from an end-user perspective. To this end, we conducted a large-scale Amazon Mechanical Turk study comparing the popular state-of-the-art explanation methods to determine which are better for explaining model decisions empirically. Our results show that Explanation by examples was the most preferred type of Explanation. We also offer an open-source library *ExMatchina*, providing a readily available and widely applicable implementation of explanation-by-examples. Then, we focus on interpretable DNN models, especially models that provide concept-based explanations. We proposed *CoDEx*, an automatic Concept Discovery and Extraction module that identifies a rich set of complex concepts from natural language explanations of videos–obviating the need to predefine the amorphous set of concepts. Finally, we introduce *XCHAR*, an Explainable Complex Human Activity Recognition model that accurately predicts complex activities and provides explanations in the form of human-understandable temporal concepts.

The dissertation of Jeya Vikranth Jeyakumar is approved.

Anthony John Nowatzki

Jonathan Chau-Yan Kao

Xiang Chen

Puneet Gupta

Mani B. Srivastava, Committee Chair

University of California, Los Angeles

2022

To my parents . . .

for their endless love, support and encouragement

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This dissertation is the culmination of many years of hard work. It would not have been possible without the encouragement, support, and advice from several people at and outside UCLA. I would like to express my gratitude to my family, professors, friends, and everyone else who assisted me in completing my Ph.D., both professionally and personally.

First and foremost, I'd like to express my gratitude to Mani Srivastava, my graduate advisor, for his assistance and support. His innovative ideas, strong awareness of technologies, and meticulous attention to detail have had a significant impact on the direction of my study. I am grateful for all of his time, energy, and ideas in making my Ph.D. research meaningful, fruitful, and pleasant. As a successful scholar and devoted lecturer, he is unquestionably a role model for me. I will be eternally grateful for his unwavering support in my studies and personal life. I would like to thank my thesis committee members, Professor Puneet Gupta, Professor Jonathon Kao, Professor Anthony Chen, and Professor Toni Nowawtzki, for their time and effort for assisting me in improving my research work and this dissertation.

Almost all of my research projects were collaborative, and I'd like to express my gratitude to everyone who helped me. A massive shout out to Luis, who was a postdoc in NESL at the time, for supporting me and providing me with valuable advice through the final stages of my Ph.D. Then I would like to express my gratitude to all of my internship mentors. Lucy from Arm Inc. deserves special mention for being an excellent mentor during and after my summer internships. I would also like to express my appreciation to IBM mentors Supriyo Chakraborty and Frank Le. Multiple interactions with them provided me with invaluable knowledge and insights that greatly aided my research in Explainable AI. Finally, I would like to thank Luke and Professor Alessandra from the United Kingdom, who are outstanding researchers and great collaborators.

A sincere thanks to my former and current lab mates at NESL and the fellow officers of EGSA for being amazing friends throughout my Ph.D. journey. A big thank you to my roommates in LA for making our house seem like home and making this Ph.D. adventure enjoyable and unforgettable. Finally, I want to express my gratitude to my parents (Jeyakumar

S. and Anita J.) and my younger brother (Visshwak J.) for always being there for me and showing me everlasting love, support, and encouragement.

The research presented in this dissertation is supported in part by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001 and the National Science Foundation (NSF) under award #CNS-1822935, and by the National Institutes of Health (NIH) award #P41EB028242 for the mDOT Center. 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 NSF, the NIH, 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.

VITA

2016	B.E., Electronics and Communications Engineering, Anna University (CEG), India
2018	M.S., Electrical and Computer Engineering, UCLA
2018	Machine Learning Intern, Arm Inc.
2019	UCLA ECE Department Preliminary Exam Fellowship
2019	Research Intern, Arm Inc.
2021	Research Intern, IBM
2021	Winner of IBM's "Be(e) Innovative Challenge 2021"

PUBLICATIONS

Jeyakumar, J.V., Sarker, A., Garcia, L.A., and Srivastava, M., 2022. X-CHAR: Explainable Complex Human Activity Recognition via Temporal Concepts (Under Submission)

Jeyakumar, J.V., Dickens, L., Cheng, Y.H., Noor, J., Garcia, L.A., Echavarria, D.R., Russo, A., Kaplan, L.M. and Srivastava, M., 2022. Automatic Concept Extraction for Concept Bottleneck-based Video Classification. (Under Submission)

Lam, J., Quan, P., Xu, J., **Jeyakumar, J.V**. and Srivastava, M., 2020, November. Hard-Label Black-Box Adversarial Attack on Deep Electrocardiogram Classifier. In Proceedings of

the 1st ACM International Workshop on Security and Safety for Intelligent Cyber-Physical Systems (pp. 6-12).

Jeyakumar, J.V., Noor, J., Cheng, Y.H., Garcia, L. and Srivastava, M., 2020. How can i explain this to you? an empirical study of deep neural network explanation methods. Advances in Neural Information Processing Systems, 33, pp.4211-4222.

Jeyakumar, J.V., Cherkasova, L., Lajevardi, S., Allan, M., Zhao, Y., Fry, J. and Srivastava, M., 2020, September. Combining Individual and Joint Networking Behavior for Intelligent IoT Analytics. In International Conference on Internet of Things (pp. 45-62). Springer, Cham.

Jeyakumar, J.V., Lai, L., Suda, N. and Srivastava, M., 2019, November. SenseHAR: a robust virtual activity sensor for smartphones and wearables. In Proceedings of the 17th Conference on Embedded Networked Sensor Systems (pp. 15-28).

Jeyakumar, J.V., Lee, E.S., Xia, Z., Sandha, S.S., Tausik, N. and Srivastava, M., 2018, October. Deep convolutional bidirectional LSTM based transportation mode recognition. In Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers (pp. 1606-1615).

Lee, E.S., **Jeyakumar, J.V.**, Balaji, B., Wilson, R.P. and Srivastava, M., 2017, November. AquaMote: Ultra low power sensor tag for animal localization and fine motion tracking. In Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems (pp. 1-2).

CHAPTER 1

Introduction

Today, smart devices such as mobile phones, smart IoT devices, and wearables are becoming truly ubiquitous. These devices are equipped with multiple sensors [LML10] including cameras, Radiofrequency (RF) sensors, and Inertial measurement units (IMUs) (accelerometers, gyroscopes, and magnetometers) that enable the devices to infer the activities of a person. The sensors on these devices generate a constant stream of data, providing a wealth of information about the behavior of their users. Furthermore, as machine learning technology continues to improve, these devices help support the development of smart cities, smart homes, increased automation, better healthcare, and more connectivity across the globe.

In various disciplines, information about the same phenomenon can be acquired from different types of detectors, at different conditions, in multiple experiments, or subjects, among others. We use the term "modality" for each type of acquisition sensor. Due to the rich characteristics of natural phenomena, a single modality rarely provides complete knowledge of the phenomenon of interest. To ensure the consistent detection of activities, multiple devices with different types of sensors are usually employed, and they are called multimodal data sources. The increasing availability of several modalities reporting on the same system introduces new degrees of freedom, which raise questions beyond those related to exploiting each modality separately. To exploit the advantages of the multiple sensors, it is necessary to fuse the data delivered by these sensors. The fused data should contain information from all the sensors and ensure a higher certainty about the detection of the activities performed.

By virtue of their design architecture, Deep Neural Networks (DNNs) have been shown to approximate arbitrary functions, mapping inputs to outputs successfully. This has resulted

in DNNs being employed to fuse this multimodal data and achieve super-human level performance on various complex tasks like Human Activity Recognition (HAR), cancer prediction, anomaly detection, computer vision, language translation, etc. However, with the increase in the adoption of deep learning for sensor systems, new challenges have arisen. First is the heterogeneity of devices. Not every device is the same and therefore models trained on the data from one device cannot be easily transferred to work on another device. Secondly, Deep Learning models used on this multimodal data to make important predictions in critical contexts are black-box by nature. Therefore, explanations supporting the output of a model are crucial, e.g., in precision medicine, where experts require far more information from the model than a simple binary prediction for supporting their diagnosis. For doctors, this helps monitor patients' routines and lifestyles to provide better treatment. Other examples include autonomous vehicles in transportation, security, and finance. Hence, we need a framework to make predictions that is robust to the heterogeneity of the devices and provide trustworthy predictions accompanied by human-understandable explanations.

1.1 Challenge 1: Heterogeneity in Devices

With the advancement in hardware and software technologies every year, industry and research develop thousands of devices with different capabilities, which creates even more heterogeneity among devices. The three main types of heterogeneity are:

Set of sensors: Each device has its own set of sensors [LJB17]. For example, unlike surveillance cameras that only capture RGB images, the cameras on autonomous cars capture depth information as well. Similarly, all smartphones and wearables do not have the same set of inertial sensors. The inertial measurement unit can have only an accelerometer or accelerometer and gyroscope, or accelerometer, gyroscope, and magnetometer. Wearables usually are equipped with only an accelerometer and do not have a gyroscope and magnetometer to reduce energy consumption.

Sampling frequency: Based on the cost, memory and energy consumption, each device is configured to capture data at varying sampling frequencies. For example, the cameras used

in sports can record videos at more than 120 frames per second (fps), whereas surveillance cameras that are running 24x7 record videos at 10-30 fps to save the storage capacity. Similarly, smartphones can collect inertial sensor readings in the 50 - 200 Hz frequency range. In contrast, wearables like smartwatches and health bands collect inertial sensor readings at a much lower frequency of 10-100Hz as low frequency helps in increasing their battery life.

Location of devices: Not all devices are placed in the same location, and sometimes the location of the devices varies over time. For example, a person might own multiple smart devices, usually located in different locations on the body. (E.g., Smartphones in the pockets, smartwatches in the wrists and smart shoes on the feet.) Also, the person might keep his smartphone in their left or right pockets, which cannot be controlled.

1.2 Challenge 2: The Need for Explainability

The increased adoption of deep learning-based solutions, often into mission-critical systems, has accelerated the need to open up these opaque DNNs and explain the inner workings leading to their decision [BCR97, DK17]. In human-machine hybrid systems, a human-understandable explanation accompanying the DNN output allows smooth interfacing between the human decision-makers and their machine counterparts. For example, "Robot Radiologists" now provide superior MRI and X-Ray image classification compared to the average trained human expert [Rea19]. A life-or-death diagnosis undeniably justifies using the best-performing model; however, it is unreasonable for either a patient or medical professional to accept an automated prediction at face value. Also, privacy regulations such as GDPR mandate the "right to explanation" as a privilege of the content owner, which makes explanation not only desirable but also a necessity [gdp18b]. Explanations are similarly essential in the coalition military domain because successful battlefield decision-making based on situational understanding produced by machines depends not only on the quality of inferences but also on providing the human decision-maker with adequate explanations to establish trust and collaboration.

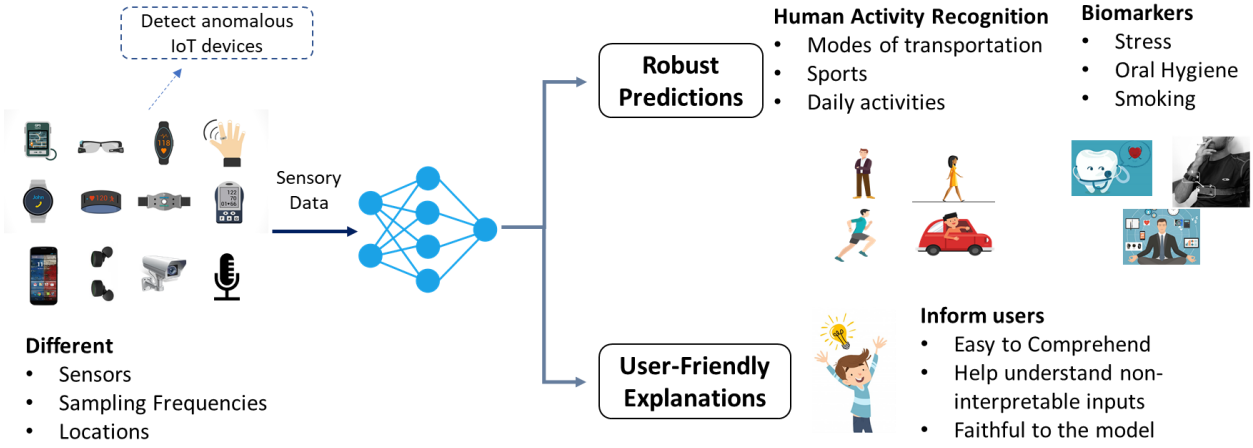


Figure 1.1: The Vision: To provide robust predictions and human friendly explanations for classification tasks

1.3 The Vision: Robust and Interpretable predictions

In this thesis, we design deep learning models that are robust to the heterogeneity of devices. We also understand the human preferred explanation methods and design interpretable deep learning models that can explain a prediction in a human-friendly manner across multiple modalities, including inputs like motion sensors which are inherently difficult to understand. Figure 1.1 shows the overall vision of this dissertation.

1.4 Dissertation outline

The main contributions of the dissertation are organized in different chapters as follows:

- Chapter 2 proposes *SenseHAR*, a robust virtual activity sensor that maps the raw sensor values to a shared low dimensional latent space and is invariant to the device heterogeneity.

- In Chapter 3, to solve the scalability issues with managing the millions of devices, we designed and evaluated *IoTelligent*, a method based on: (i) autoencoders, which extract the relevant features automatically from the network traffic stream; (ii) DBSCAN clustering to identify the group of devices that exhibit similar behavior, to flag anomalous

devices.

- Chapter 4 presents and discusses the results of a Mechanical Turk study identifying the relative preference of post-hoc explanation methods by an average non-technical end-user. It also offers an open-source library, ExMatchina, providing a readily available and widely applicable implementation of explanation-by-example.

- In Chapter 5, we develop *CoDEx*, a concept discovery and extraction pipeline that leverages NLP techniques to automatically extract complex concept abstractions from crowd-sourced, natural language explanations for a given video and label–obviating the need to define a necessary and sufficient set of concepts manually

- Chapter 6 introduces *XCHAR*, an interpretable DNN model for explainable complex human activity recognition that achieves state-of-the-art accuracy and provides human-understandable explanations in the form of temporal concepts.

- Finally, Chapter 7 provides the possible future directions of our research and Chapter 8 concludes this dissertation.

CHAPTER 2

SenseHAR: A Robust Virtual Activity Sensor for Smartphones and Wearables

Modern smartphones and smartwatches are equipped with inertial sensors (accelerometer, gyroscope, and magnetometer) that can be used for Human Activity Recognition (HAR) to infer tasks such as daily activities, transportation modes and, gestures. HAR requires collecting raw inertial sensor values and training a machine learning model on the collected data. The challenge in this approach is that the models are trained for specific devices and device configurations whereas, in reality, the set of devices carried by a person may vary over time. Ideally, one would like activity inferencing to be robust of this variation and provide accurate predictions by making opportunistic use of information from available devices. Moreover, the devices may be located at different parts of the body (e.g. pocket, left and right wrist), may have different sets of sensors (e.g. a smartwatch may not have gyroscope while a smartphone might), and may differ in sampling frequencies. In this paper, we provide a solution which makes use of the information from available devices while being robust to their variations. Instead of training an end-to-end model for every permutation of device combinations and configurations, we propose a scalable deep learning based solution in which each device learns its own sensor fusion model that maps the raw sensor values to a shared low dimensional latent space which we call the 'SenseHAR'-a virtual activity sensor. The virtual sensor has the same format and similar behavior regardless of the subset of devices, sensor's availability, sampling rate, or a device's location. This would help machine learning engineers to develop their application specific (e.g., from gesture recognition to activities of daily life) models in a hardware-agnostic manner based on this virtual activity sensor. Our evaluations show that an application model trained on SenseHAR achieves the state of the

art accuracies of 95.32%, 74.22% and 93.13% on PAMAP2, Opportunity(gestures) and our collected datasets respectively.

2.1 Introduction

In the past decade, the market surrounding smartphones and wearable technology has grown exponentially [Sta18]. Smartphones and other personal wearables used for fitness and health monitoring are cheap, ubiquitous and a survey shows that they are owned by more than 85% and 20% of the United States population [Del18, IDC18] respectively. These devices are equipped with multiple sensors [LML10] and Inertial measurement units (IMUs) are the most prominent sensors that have a combination of microelectromechanical (MEMS) accelerometers, gyroscopes, and magnetometers that enable the devices to infer the activities of a person. While, prior research identifies smartphones as the widely used sensing modality in HAR [MHL15, RMB10], smartwatches are also being utilized extensively to infer diverse gestures for various applications such as sign language [BSL03] and medical rehabilitation [BKB14]. Today most people own multiple smart devices such as smartphones, smartwatch, fitness bands, smart shoes, smart belts, etc. and we should be able to leverage on the information from these multiple devices to improve the accuracy of activity recognition. This process of combining information from multiple devices and sensors is called Multimodal sensor fusion. There exist multiple ways to do multimodal sensor fusion for HAR. Previous studies suggest feature concatenation of the sensor data [BWG12, HNT13, NMN15], an ensemble of models [KP05, ZSF07] for multimodal sensor fusion and propose frameworks for training data from multiple inertial sensors.

But the existing approaches fail to take into consideration the heterogeneity of the devices. Stisen et al. [SBB15] explain that all devices are different and vary in their hardware and operating systems. They can have a different number of sensors, different sampling frequencies and be located at different locations on the body. Research by Amft [Amf10] and Blunck et al. [BBF13] show that when these HAR systems are deployed in real life, i.e., across heterogeneous devices and usage situations, recognition performances are often significantly

lower than what is suggested in research. Furthermore, the existing methods for HAR assume that all the sensing devices exist at all times to make the inference and but in reality, the set of devices carried by a person tends to vary over time even if they own multiple devices.

Unlike the existing solutions, we need a machine learning framework to exploit the information from the available sensors at any given time to give a constant stream of correct activity predictions. Hence, we propose a solution that makes opportunistic use of the information from available devices while being robust to their variations. We provide a scalable deep learning based framework in which each device learns its own sensor fusion model that maps the raw inertial sensor values to a shared low dimensional latent space which we call the 'SenseHAR'- virtual activity sensor. SenseHAR is robust to the heterogeneity in devices,i.e, it has the same format and similar behavior regardless of the subset of devices, sensor's availability, sampling rate, or a device's location. Since the devices get mapped to the same latent space, we combine information from the available devices by aggregating the SenseHAR values from each device. SenseHAR decouples the hardware from software and helps researchers to train their application specific (e.g., from gesture recognition to activities of daily life) algorithms in a hardware-agnostic way. In other words, the same application model trained on SenseHAR values for daily activity recognition can be used with any device or any combination of devices. Finally, we evaluate our method on the PAMAP2, Opportunity and our collected Multi-Modal Activity Recognition dataset (MMAR) which have data from multiple wearables containing inertial measurement units and located in different parts of the body. We compare the performance and find that our solution to train an application specific model using SenseHAR achieves the state-of-the-art accuracy and f1-score when compared with the end-to-end models trained from scratch on raw sensor data.

To summarize, our main contributions of the paper are as follows:

- We propose a Sensor Fusion model for each device that maps the raw sensor values to a shared low dimensional latent space- SenseHAR, the virtual activity sensor.
- We implement a training framework that enables each device to learn the proposed

sensor fusion model and to obtain a robust SenseHAR that is invariant to the device heterogeneity.

- We provide a method to calibrate SenseHAR in a new device or hardware setup with the help of the SenseHAR from the pre-trained device.

- We perform thorough evaluations on PAMAP2, Opportunity and on our collected multimodal dataset (MMAR) to show that the SenseHAR captures maximum information from all the sensors from the different available devices and hence can be used to train models for various HAR applications.

2.2 Background

2.2.1 Human Activity Recognition

The broad field of study to classify sequences of sensor data into known well-defined movements or actions of a person is called Human Activity Recognition(HAR). Prior research has indicated that smartphones and wearables as the widely used modalities in sensor-based HAR [ITU15, SBI16] that can infer numerous activities including where they go, when they exercise [MHL15] and how well they sleep [CLC13]. HAR is of great importance because it encourages users to adopt a healthier lifestyle by increasing personal awareness about physical activities and its positive consequences on health. It also has a great significance in designing artificially intelligent human-computer interface [FPB08] for various applications such as sign language [BSL03] and medical rehabilitation [BKB14]. While accelerometer is the most prominent sensor used in activity detection [VBD96, ALK10, SKK14], gyroscope and magnetometer [LSH09, AB10] are also integrated to improve the probability of correct predictions and to identify activities such as the mode of locomotion [HNT13, JLX18, RMB10, WGM18], posture [YLM13], gait [ITU15], fall detection [WAP13], exercise [SBV11] and sleep pattern [NDE12]. Sensor data from smartwatches are utilized to infer different gestures such as finger movements [WRD16], brushing habits [ASS19], smoking habits [PCC14], eating [SBI16, DSW14] and cooking [ZS12].

2.2.2 Multimodal Sensor Fusion

In HAR, to ensure the consistent detection of activities and also to recognize fine gestures multiple devices with different types of sensors are usually employed and they are called multi-modal data sources. To exploit the advantages of the multiple sensors, it is necessary to fuse the data delivered by these sensors. The fused data should contain information from all the sensors and ensure a higher certainty about the detection of the activities performed. Sensor fusion is an existing concept and is widely applied in the field of computer vision (merging video with audio signals) [NKK11, WJW16], in modern cars [GWS11, Hec18], medical EEG devices [HZL16, ALB18] and robotics to detect the existence of objects and free space [SGA17, NR15], which is used by mobile robots for obstacle detection or path planning. The fusion of these sensor data, therefore, plays a very crucial aspect of the perception process in almost all automated, assistance and autonomous systems. Since this fusion is be based on a time series of data from several different sensing modalities, it is called multi-modal sensor fusion. Multi-modal sensor fusion offers advantages in terms of being able to sense various complementary aspects of an object with the different modalities, i.e. it increases the information gained by the system.

2.2.2.1 Traditional Machine Learning

In the traditional method, each channel or axis of each sensor is considered as an individual sequence. Then, for each input sequence, the list of features in both time and frequency domain are extracted from the raw data. The frequency domain features and Time domain features are explained in detail in the prior works [EAA16, Dar09]. These features are just taken together and classical machine learning models like SVM, Random forests and XGBoost or sequence-based models like Hidden Markov models and Conditional Random Fields [AGO12, CPR11, LC11, VVL07] are applied to make the inferences on the activities.

2.2.2.2 Deep Learning

With the advent of deep learning more sophisticated algorithms emerged to make predictions from multiple sensor streams. S.Yao et al. [YHZ17] suggest a deep learning framework by exploiting local interactions within each sensing modality. They apply Fourier transform to obtain better local frequency patterns, then merge local interactions of different sensing modalities into global interactions and extract temporal relationships. But recent research has shown that we don't have to extract specific features from the raw sensor values anymore as the deep neural networks are capable of learning that by themselves. There have been works using CNNs [ZNY14], LSTM [CZZ16] and hybrid models like Convolutions LSTMS [JLX18] to make the activity predictions. The existing solutions to perform deep learning based multimodal sensor fusion are as follows:

Feature Concatenation: This is a straightforward technique that combines multimodal information in the feature level. Data obtained from different modality data are firstly concatenated into a long feature vector. For example, Bulling et al. [BWG12] combined the features of accelerometers and gyroscopes. Similarly, combining accelerometers with physiological sensors [YC08], microphone [LWJ04], or location sensors [PEA07] is also investigated to improve HAR performance. A few drawbacks of this approach is that these methods have the tendency to neglect the intra-sensor correlations as stated in [RTB18]. Sometimes it's not possible to perform concatenation at feature level because of the difference in the sampling rates of the modalities and the general procedure to mitigate this is to resample the data to the required frequency.

An ensemble of classifiers: This is another commonly adopted method in multimodal activity models [KP05, ZSF07]. Ensemble classifiers make the final prediction by combining the predictions from multiple base models and the common techniques used are bagging, boosting and stacking. Prior research generally confirms that a diverse set of models increases predictive accuracy in comparison to single models combination [Fin11, PEA10]. However, a fundamental weakness of EC as stated in [RTB18] is that because fusion takes place at the end of the model, a lot of potential information and cross-sensor relationships are already

lost.

Modality-Specific Architectures: Recent work by Radu et al. [RTB18] proposes Modality-Specific Architecture (MA) in Deep Learning which comprises of two types of hidden layers – hidden layers related to a specific sensor type and hidden layers that capture unified concepts across the sensor. Separate architectures are built for each modality to first learn sensor-specific information before their generated concepts are unified through representations that bridge across all the sensors (i.e., shared modality representations) later in the network.

2.2.3 Heterogeneity in smartphones and wearables

Set of sensors: All smartphones and wearables do not have the same set of inertial sensors. The inertial measurement unit can have only an accelerometer (3 DOF) or accelerometer and gyroscope (6 DOF), or accelerometer, gyroscope, and magnetometer (9 DOF). Wearables usually are equipped with only an accelerometer and don't have gyroscope and magnetometer to reduce energy consumption.

Sampling frequency: The sampling rates of the inertial sensors in different devices are different. Smartphones can collect inertial sensor readings in the frequency range of 50 - 200 Hz. On the other hand, wearables like smartwatches and health bands collect inertial sensor readings at a much lower frequency of 10-100Hz as low frequency helps in increasing their battery life.

Location of devices: A person might own multiple smart devices and they are usually located in different locations on the body. (E.g., Smartphones in the pockets, smartwatches in the wrists and smart shoes in the feet.)

2.3 Related Works

In deep learning, before we use a neural network for a task (classification, regression), the usual architecture is to extract features through many layers (convolutional, recurrent, pooling, etc.). These layers map the inputs to the latent space on which the last classification layer is applied.

Latent space has gained a lot of importance in Natural Language Processing(NLP) for word embeddings [GL14], [PSM14], in social network analysis for random graph models [HRH02] and in computer vision for applications like style transfer [ZPI17], data compression [HS06] and data interpolation [WZX16]. The work by Ngiam et al. [NKK11] shows how to learn a shared representation between modalities and evaluate it on a unique task, where the classifier is trained with audio-only data but tested with video-only data and vice-versa.

For cross-domain activity recognition, the concept of transfer learning where certain parameters can be transferred while training a new model for a different application has been explored [MR16, CFK13] but this method still requires true activity labels to train the new model. Chen et al. [CWH19] introduce stratified transfer learning approach that uses pseudo labels from existing sensors instead of true labels. It performs intra-class knowledge transfer between domains iteratively to transform them into the same subspaces. Rokni et al. [RG18] introduce an autonomous training method where, pseudo labels obtained from existing sensors are concatenated with features from the new sensor and then clustered to improve accuracy. But these approaches only work on the traditional method of extracting features from the raw sensor values. Xing et al. [XSB18] show that the shared latent space representation exists for time series sensory data, and it can help transfer knowledge from ambiance edge devices to wearable edge devices and vice versa. Radu et al.'s deep learning based modality specific architecture [RTB18] does sensor fusion in latent space. But this work does not take into consideration the heterogeneities in the devices and sensors and hence, a new end-to-end model has to be trained for each device and application. Also, these models assume that all the modalities are present at any given time which is generally not the case in real life implementations. The work by Khan et al. [KRM18] assumes that the distribution of weights and biases in the convolutional layers remains largely unchanged across different activities, and thus automatically adapts and learns the model across different domains with minimal labeled data. This approach does not address the problem of a new device having a different hardware configuration than the existing device. The conventional procedure to take care of the difference in sampling rates of the devices is to interpolate or resample the collected data to the required frequency. The two common methods of interpolation used for HAR are linear

[CSC13] and cubic spline [TIH07]. This means additional preprocessing has to be done before feeding it to the HAR models. Stisen et al. [SBB15] propose a clustering-based approach as a mitigation technique to improve HAR performance in the presence of heterogeneities but their work is limited to devices with a single modality(accelerometer).

One of the early efforts in creating a virtual sensor from sensor fusion was the work on virtual gyroscope [CXQ08] which proposed the theory of gyro-free IMU. The virtual gyroscope with multigyroscope and accelerometer array (MGAA)configuration merges the outputs of multi-gyroscopes and specifically placed accelerometers through a Kalman filter. The work by LiKamWa et al. [LLL13] proposed Moodscope, a virtual mood sensor for a smartphone that measures an important mental state of the user based on the interactions with the smartphone. These papers relied on traditional signal processing and decision-based algorithms and hence do not generalize well for different devices and different users.

2.4 SenseHAR: Virtual activity sensor

2.4.1 Overview

We introduce SenseHAR, a robust virtual activity sensor for wearables and smartphones and Figure 6.3 shows the overview of SenseHAR. The inertial sensor data from the available heterogeneous devices are fused using a sensor fusion network to a shared low dimensional latent space. This shared latent space is called SenseHAR- a virtual activity sensor. Fusing information from different devices to a shared latent space combines the advantages of both feature concatenation and ensemble classifiers as it captures both inter-device and intra-device correlations. SenseHAR is robust to variations in hardware configurations such as the set of sensors, sampling frequency, device location, and device combinations. Therefore, application-specific machine learning models for activity recognition can be trained on the SenseHAR values in a hardware agnostic manner. The same application model trained on SenseHAR values for daily activity recognition can be used with any device or any combination of devices. This enables to infer activities continuously even while the set of devices carried by the person

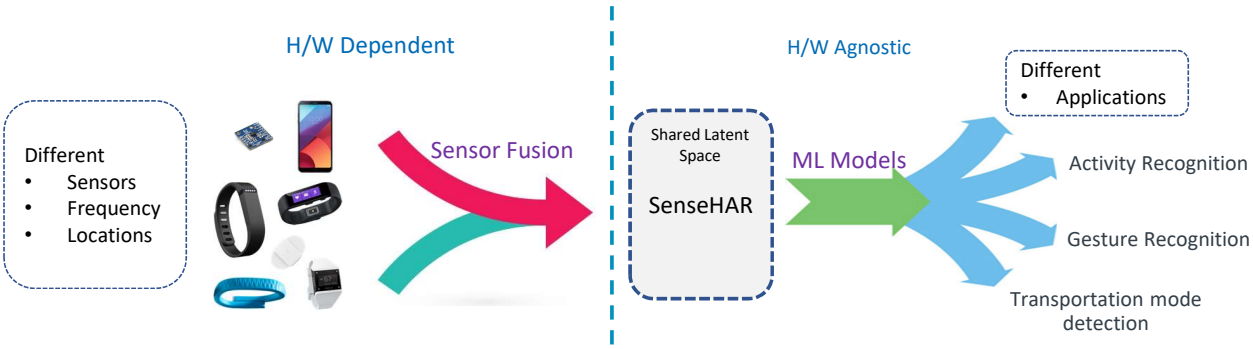


Figure 2.1: SenseHar Overview: Data from the inertial sensors of heterogeneous devices are fused together to construct a shared latent space (SenseHAR). Machine Learning models for diverse applications are trained using SenseHAR in a hardware agnostic way.

varies over time, with negligible to no loss in performance.

2.4.2 Requirements

An Ideal virtual activity sensor should have the following requirements:

- Similar format and behavior regardless of the number of available devices.
- Generate values at a constant frequency irrespective of the sampling rates of the individual devices.
- Provide all the required information for activity detection so that it can be used to train models for different applications like identifying gestures and detecting daily activities.
- Have only a few channels. In our paper, the virtual activity sensor has three channels similar to the axes in the IMU sensors and the reason for restricting to only three dimensions is explained in section 2.6.2 Capture the correlation across different sensors and at different locations

2.4.3 Training framework

A naive way to construct SenseHAR would be to use convolutional layers on the sensor data from each device to extract features in the latent space but this would not result in a shared

common latent space for all devices and would be specific to each device. Instead, we propose a branched, parallel training architecture to obtain robust SenseHAR. Figure 2.2 shows the training framework. Let 'n' be the total number of devices and $\mathcal{D}$ refers to the set of available devices $\mathcal{D} = \{D_i\}, i \in \{1, ..., n\}$. Let 's' be the number of sensors in each device and it can have the values {1,2,3}. Eg. The value of 's' is '1' if the IMU has only an accelerometer, '2' if accelerometer and gyroscope or '3' if accelerometer, gyroscope, and magnetometer. $\mathcal{F}$ refers to the set of Sensor fusion models for each device. $\mathcal{F} = \{F_i\}, i \in \{1, ..., n\}$. The first part of the training framework is the sensor fusion network which consists of 'n' Sensor Fusion models corresponding to each device. Let $\mathcal{B}$ be the set of branches in the training framework that follows the sensor fusion network. There are a total of 'n+1' branches $\mathcal{B} = \{b_i\}, i \in \{1, ..., n+1\}$, one for each device and one branch for the combination of all devices. Each branch is mapped to the common shared latent space–SenseHAR. The final 'n+1'$^{\text{th}}$ branch aggregates the output of the sensor fusion model from all the individual devices by taking the arithmetic mean(μ). This helps in constraining the values of SenseHAR to lie within the same latent space while combining the information from multiple devices. All the branches share the same application model 'M-shared'. The models are trained using the activity labels by minimizing the sum of the cross-entropy loss of all the branches. The loss function is given in Equation 3.1 where 'm' is the total training samples, 'b' the number of branches, 'c' is the total number of output classes, 'y' is the true label and 'a' is the activation of the output layer. This method of training constructs a shared latent space between the Sensor Network and Application model, which contains the high-level features from the sensor streams of all devices making it robust and invariant to heterogeneity.

$$Loss(L) = \frac{1}{m} \sum_{i=1}^{m} \sum_{b=1}^{n+1} \left(\log \sum_{j=1}^{c} e^{a_{bj}(\mathbf{x})} - a_{by(i)}(\mathbf{x}^{(i)}) \right) \tag{2.1}$$

2.4.4 Sensor fusion network

The sensor fusion network comprises the sensor fusion model of each individual device as shown in Figure 2.2. This network is responsible for mapping the raw sensor streams to

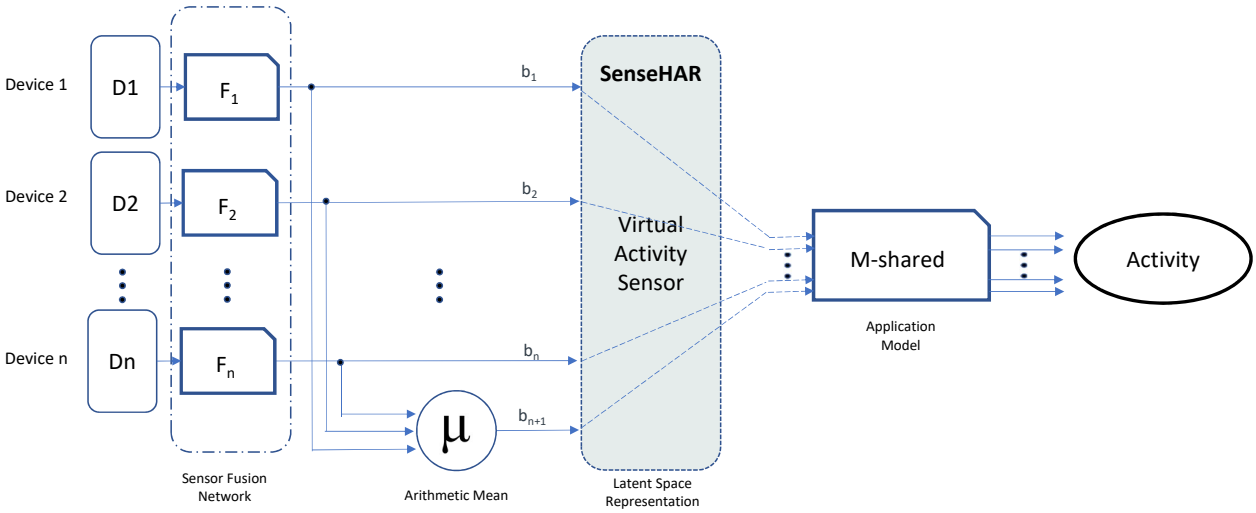


Figure 2.2: Branched Training Framework to construct SenseHAR: It has (a)Sensor Fusion Network comprising 'n' Sensor fusion models which maps to the shared latent space 'SenseHAR' (b) Shared Application model to predict activities.

the shared low dimensional latent space. Each sensor fusion model consists of three stages. Figures 2.3, 2.4, 2.5 show the three stages of the sensor fusion model.

2.4.4.1 Stage 1

Each sensor stream is considered separately and passed through a series of one-dimensional convolutional layers (1-D Conv). The 1-D convolutional layer is used for extracting local features from 1D patches in every sensor sequence and can identify local patterns within the window of convolution. And because the same transformation is applied on every patch identified by the window, a pattern learned at one position can also be recognized at a different position, making 1D convolution layers translation invariant. Figure 2.3 shows the data-flow in stage 1. The input to the first stage is the data from available sensors and is of the shape [(s*3) x t] where '3' denotes the 3 axes for each sensor and 's' is the number of sensors in each device. The output of the first stage is a collection of feature maps corresponding to each filter for every sequence and is of the shape [(s *3) x t x k_1] where 'k_1' is the number of 1-D kernels.

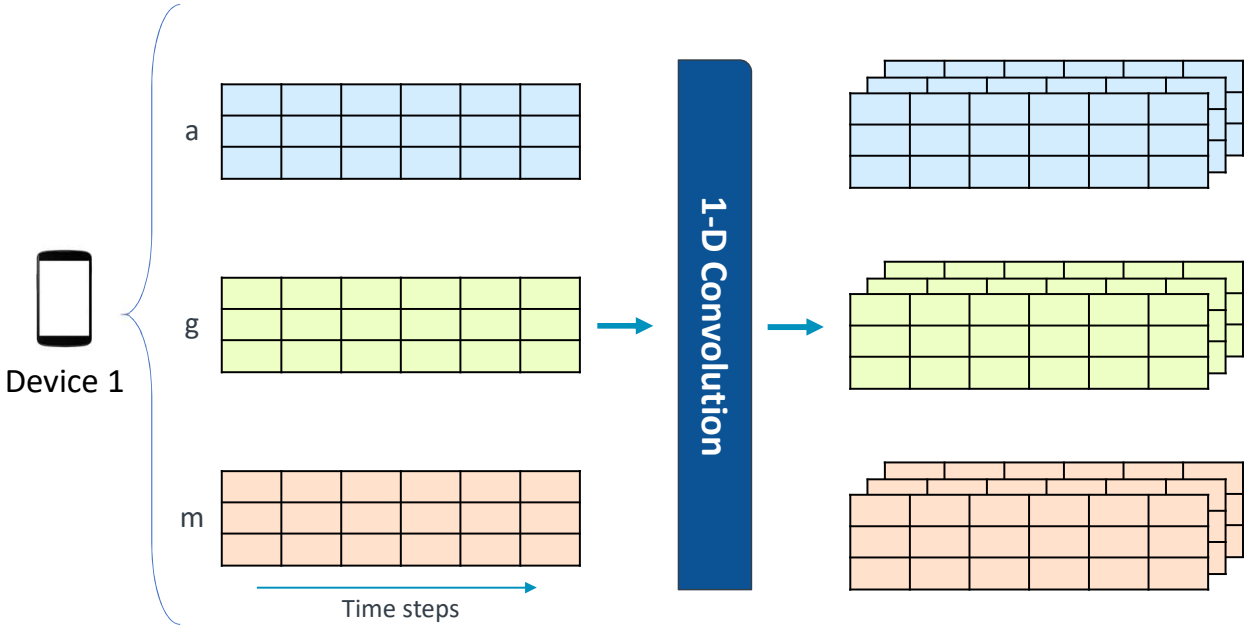


Figure 2.3: Sensor Fusion Model - Stage 1: Contains 1-D Conv layer to extract features from each sensor stream

2.4.4.2 Stage 2

The second stage is of the sensor fusion network is to capture the correlation across the corresponding axes in different sensors. So we concatenate the feature maps to the shape [3 x t x (s * k1)] and feed it to the second set of convolutional layers. The output of the second convolutional layer is of the shape [3 x t x k2] where 'k_2' is the number of convolution kernels in the 2nd stage. Figure 2.4 shows the data-flow in stage 2.

2.4.4.3 Stage 3

The initial layer of this stage is a K-Max pooling layer [KGB14]. K-max pooling over a linear sequence of values returns a subsequence of k maximum values in the sequence, instead of the single maximum value and it helps to sample down different length vectors into a fixed length. The output from the pooling layer is fed to a Time-Distributed Dense layer which applies the same Dense (fully-connected) operation to every time-step of a tensor, i.e., it allows you to apply that Dense function across every input over time. Thus the output of this layer is a sequence of the same length as the input sequence. In our paper, we use a

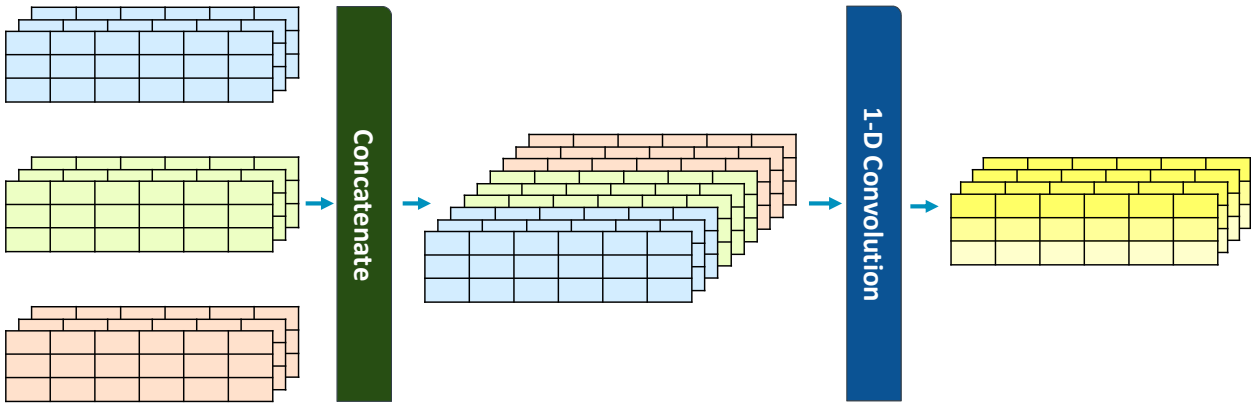


Figure 2.4: Sensor Fusion Model - Stage 2: Captures the correlation across the corresponding axes in different sensors.

Time-distributed Dense layer with 3 neurons to capture the correlation across all the feature maps and finally we get a sequence of values of constant dimension [3 x t]. The reason for choosing 3 can be understood from section 2.6.2 and Figure 2.5. Intuitively it corresponds to the 3 virtual axes of the virtual sensor.

2.4.5 Application model

The second part of the training framework is the application model(M-Shared). This application model is shared by all the branches in the training framework. It includes two stacked LSTM layers followed by an output dense layer to give the activity predictions. The LSTM layers capture the temporal information from the SenseHAR sequence. Dense layer has 'c' neurons corresponding to the number of output classes. And since this is a classification problem we use softmax (σ) as the activation of the final dense layer which is given in Equation 3.2. Softmax function is used to impart probabilities to the logits 'a' when we have multiple classes and we get the probability distribution of output classes. We consider the most probable occurrence with respect to other outputs as the predicted class. Figure 2.6 shows the application model architecture.

$$\sigma(\mathbf{a})_j = \frac{e^{a_j}}{\sum_{k=1}^{c} e^{a_k}} \tag{2.2}$$

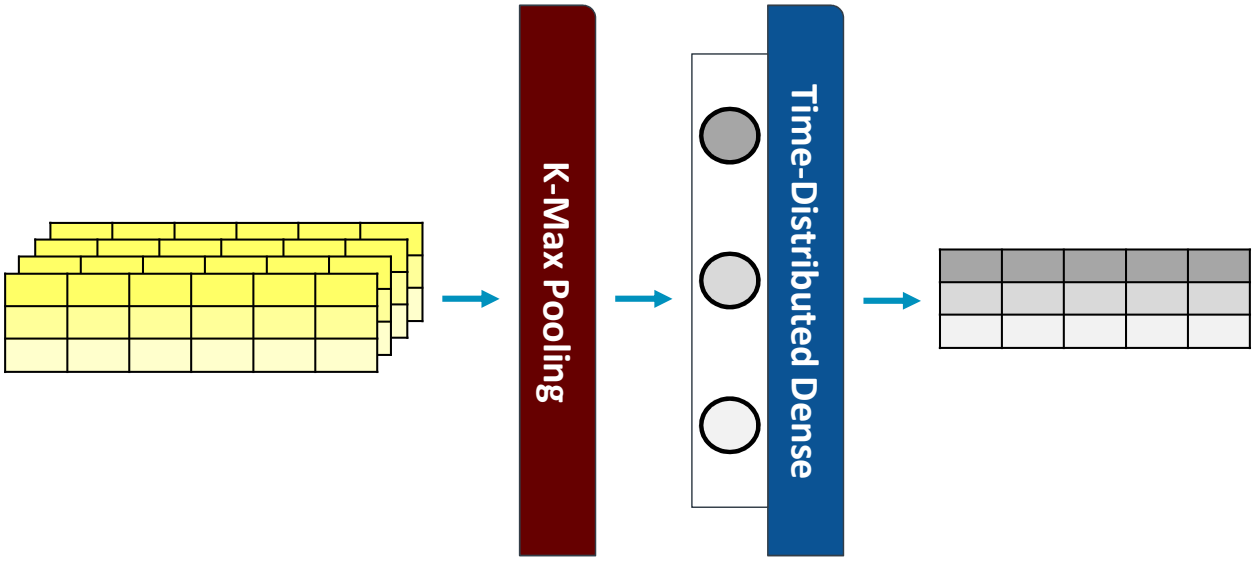


Figure 2.5: Sensor Fusion Model- Stage 3: Fuses the extracted features from the previous stages to a low-dimensional latent space

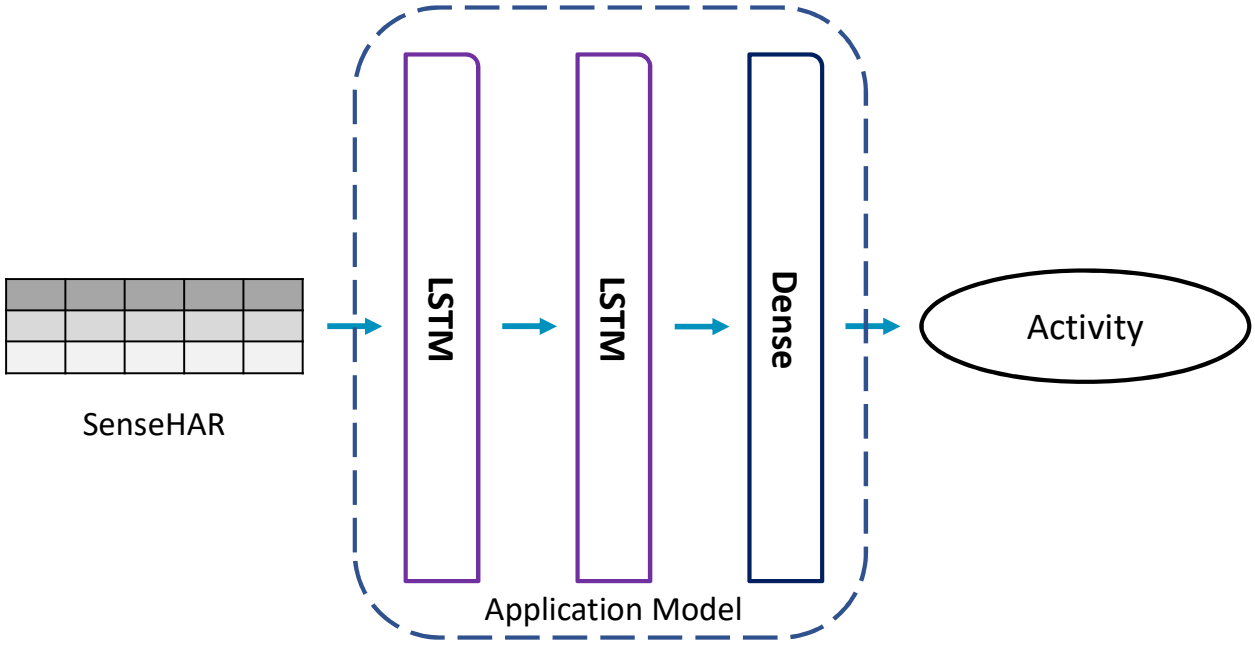


Figure 2.6: Application Model: LSTM layers to capture the temporal information and output Dense layer

2.5 Evaluation Methods

2.5.1 Constructing SenseHAR

Inertial sensor data is collected from a person wearing multiple devices while performing various daily activities. We then use the training framework discussed in section 2.4.3 to train the Sensor fusion models for each device and the Application model by training with the collected data. With the trained sensor fusion models each device can map to the shared low dimensional latent space and generate SenseHAR values.

2.5.2 Testing on Subset of devices

Though a person might own multiple devices, the set of devices carried by them varies over time. Example: Alice might own a smartphone, a Fitbit, a smartwatch and smart shoes. While going for a run she might have all of them(a smartphone in the pocket, a smartwatch on her left hand, Fitbit on her right hand and smart shoes on her feet), while she's heading to her office she might have only her smartphone and smartwatch and while reading she might just be having one of these devices. So, when Alice has only one device at a given time, we directly use the pre-trained Application model on the data from that device's SenseHAR to make inferences but when she has multiple devices, we take the arithmetic mean of the SenseHAR values from each device to combine their information and use the same pre-trained Application model on the aggregated values to infer the activities.

2.5.3 Calibrating for new Hardware

When a person buys a new device, it usually has different device hardware specifications (e.g. Different sensor sampling rate or missing the magnetometer sensor). The new device has to be calibrated such that it generates similar SenseHAR values as the existing device. To achieve that, we first need to obtain a new sensor fusion model specific for that device which has different inertial sensor modalities than that were included in the training phase. We do this by following these two steps:

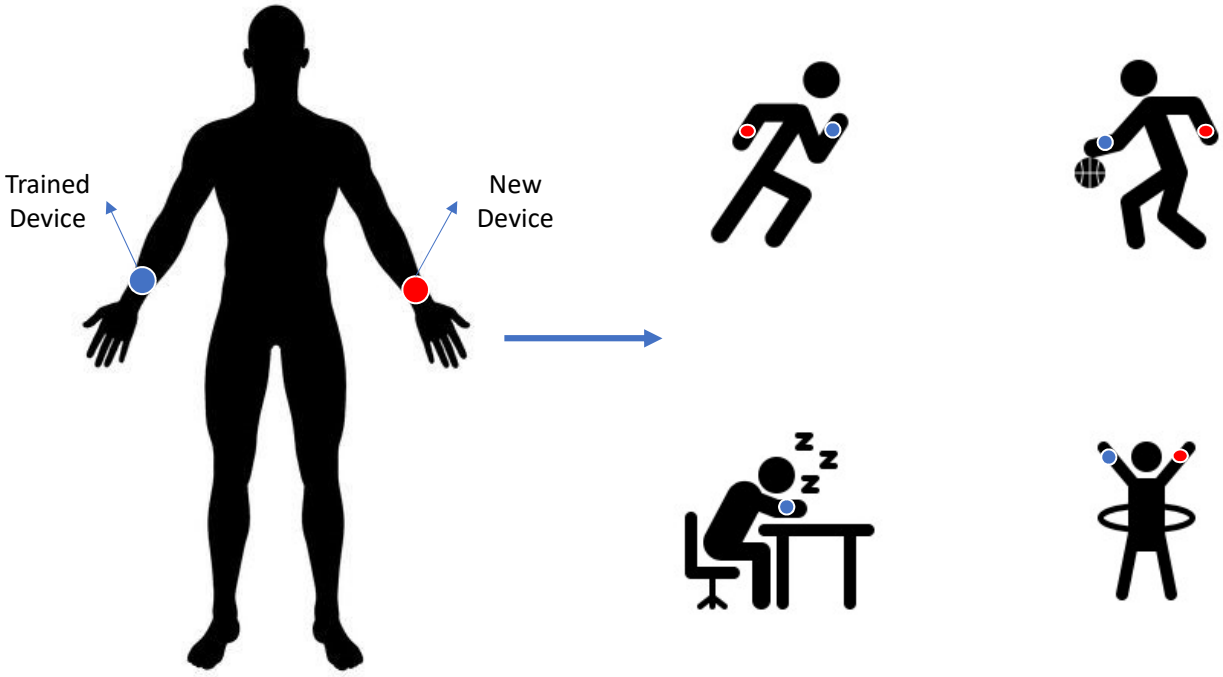


Figure 2.7: Collecting raw data from new device and SenseHAR values from pre-trained device synchronously.

2.5.3.1 Synchronous Data Collection Phase

We combine our new device or hardware setup with an existing device whose sensor fusion model is already available and synchronously collect data as shown in Figure 2.7. This provides us the raw sensor values from the new hardware and the SenseHAR values from the existing device which will be used as the ground truth soft labels.

2.5.3.2 Calibration Phase

We then initialize a new sensor fusion model for the new device which is trained to generate SenseHAR values that resemble the values obtained from the existing device by minimizing the mean squared error(regression) as shown in Figure 2.8. Once we obtain the sensor fusion model for the new device, we combine it with the same pre-trained Application model to infer the activities. This method eliminates the need to collect a large amount of labeled data for that particular hardware setup makings easier than training a new end-to-end model from the ground up.

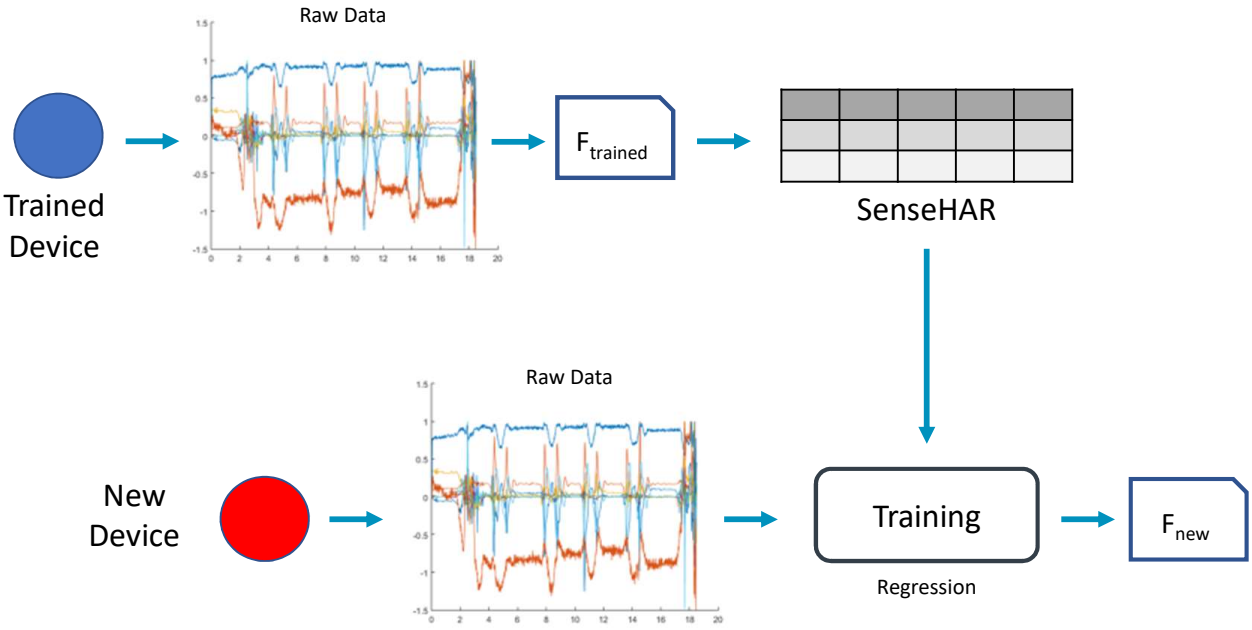


Figure 2.8: Calibrating the new sensor fusion model by training it with the SenseHAR values of the pre-trained device.

2.5.4 Training for different applications

For a given device with a trained sensor fusion model, we first collect labeled SenseHAR data for the new application and then train a new Application model for that application (Eg. gesture recognition) using the SenseHAR values and new activity labels. This method while providing similar performance as training from scratch a model on the raw sensor data has two additional benefits– 1. Saves time as compared to training a model from scratch since we are only training a few layers. 2. The application model can be designed in a hardware-agnostic way since SenseHAR has the same configuration on all devices.

2.6 Results

2.6.1 Datasets

There are plenty of publicly available datasets for human activity recognition and we selected the PAMAP2 and Opportunity datasets because they had the most number of inertial sensor modalities. We also collected our own multimodal dataset to perform more concrete

evaluations. The location of the inertial sensors in the three datasets is shown in Figure 2.9. A short description of each dataset used in this paper is as follows:

2.6.1.1 PAMAP2 [RS12b, RS12a]

This dataset had 9 subjects and each subject had to follow a particular protocol that contained 12 different activities. There were three inertial measurement units used in this study and they were placed in three different locations on the body: wrist, ankle, and chest. The IMUs contained accelerometer, gyroscope, and magnetometer and the data were collected at a sampling frequency of 100 Hz. In our evaluations, we used subject four as our test data and the remaining subjects were used as training data.

2.6.1.2 Opportunity dataset [RCR10]

This dataset had four trials and each trial had data from so many different sensors including body sensors, object sensors, and ambient sensors. Since we focus only on the inertial sensor data we consider the five inertial body-worn sensors which were worn in five different locations on the body: Left lower arm (LLA), Left upper arm (LUA), Right lower arm (RLA), Right Upper arm (RUA) and Back of the torso. These inertial units recorded accelerometer, gyroscope and magnetometer values. The activities performed had two different hierarchy of labels– High-level activities which considered the 4 major locomotion activities(sit, stand, walk, lie, random) and Low-level activities which had 17 different micro activities such as opening and closing doors, shelves, drawers and drinking tea. In our evaluation, we removed the activities with null labels and considered trial 1 to 3 as training data and trial 4 as test data.

2.6.1.3 Our Collected dataset - Multimodal Activity Recognition dataset (MMAR)

We prepared a multimodal dataset using three commonly available devices– a MotionsenseHRV wrist band worn in our left wrist that collects accelerometer and gyroscope data at a sampling frequency of 25Hz, an Android smartwatch (Asus Zenwatch 2) on right wrist that collects

Devices	Sampling Frequency	Sensors	Activities
Smartphone (Pocket)	100Hz	A,G,M	Sitting idle
			Sitting(using computer)
Smartwatch (Right wrist)	50Hz	A,G	Standing
			Walking
Wristband (Left wrist)	25Hz	A,G	Running

Table 2.1: Description of our collected Multimodal Activity Recognition (MMAR) dataset.

accelerometer and gyroscope data at 50Hz and an Android smartphone (Pixel 3)in the right pocket which records data from accelerometer, gyroscope and magnetometer at 100Hz. We collected data for the following five activities: sitting idle, sitting and using a computer, standing, walking and running. The total duration of our collected dataset was for two and a half hours, with 30 minutes approx. for each activity. The data was split into two sets such that 70% was considered for training and 30% for testing. Table 2.1 summarizes MMAR dataset.

2.6.2 Number of SenseHAR Channels

The number of channels for SenseHAR was a hyper-parameter and it depends on the number of neurons in the Time-Distributed dense layer of the sensor fusion model as discussed in section 2.4.4.3. We wanted to find the smallest number of channels that can capture information and the correlation across different sensor modalities without the loss in performance of the model. This is because the number of channels affect the model size and hence having fewer channels results in fewer parameters and a simpler model. So to find the smallest number of channels for the virtual activity sensor, we created different models having the SenseHAR channels ranging from 1 to 10 based on our sensor fusion model architecture. We evaluated this on Opportunity and PAMAP2 datasets and plotted the f1-scores achieved vs the number of channels, to obtain the graph shown in Figure 2.10. From our analysis, we found that

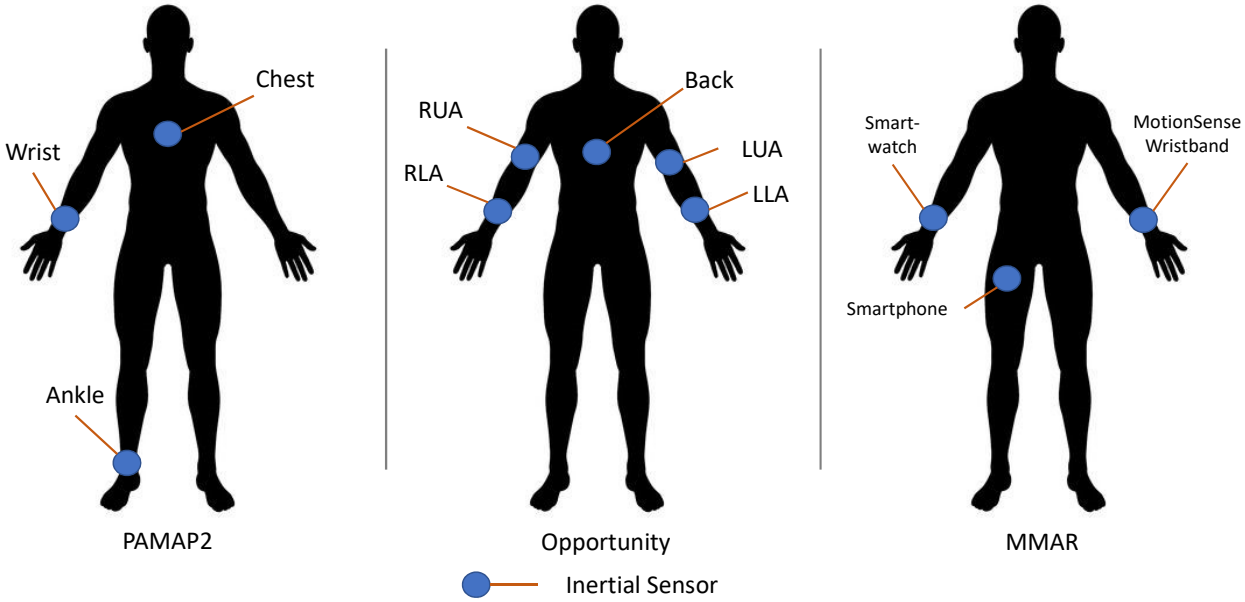


Figure 2.9: Placement of Inertial sensors in the different datasets. PAMAP2 dataset has 3, Opportunity has 5, and our collected dataset (MMAR) has 3 inertial sensors placed at different locations on the body.

increasing the number of channels beyond three did not have a significant increase in the performance of the model for both the datasets and hence we decided to use three channels to represent SenseHAR.

2.6.3 Baseline models

To evaluate the performance of our proposed method using SenseHAR we used the following existing deep learning based multi-modal architectures as our baseline models.

Feature Concatenation methods: The sensing modalities are concatenated to a single modality vector and we considered the three common deep learning architectures– Dense Neural Network(DNN), Convolutional Neural Network(CNN) and a hybrid Convolutional LSTM neural network.

Modality Specific Architecture: We implemented the two modality specific architectures MA-DNN and MA-CNN as described by Radu et al. in [RTB18] which achieved state of the art performance on multi-modal datasets. These architectures have a dedicated DNN and

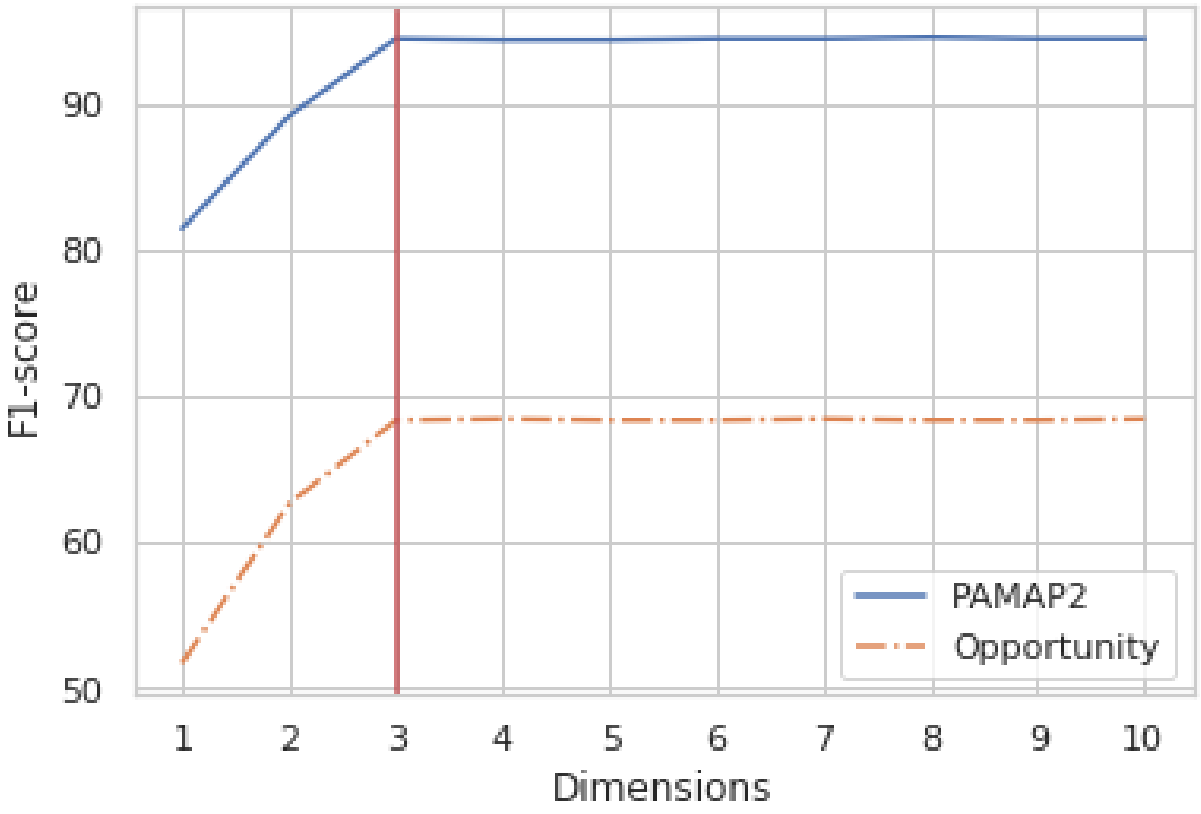


Figure 2.10: Number of SenseHAR channels vs F1-score for PAMAP2 and Opportunity datasets. There was not a significant gain in performance when the SenseHAR dimensions increased beyond three.

Datasets	DNN		CNN		Conv-LSTM		MA-DNN		MA-CNN	
	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
PAMAP2	85.82	0.8622	90.40	0.9036	94.68	0.9446	92.16	0.9204	95.14	**0.9499**
Opportunity (Gestures)	67.21	0.5842	71.13	0.6431	74.80	**0.6792**	72.86	0.6684	74.40	0.6754
Opportunity (Locomotion)	85.67	0.8494	87.25	0.8778	89.56	0.8968	88.37	0.8841	90.52	**0.9044**
MMAR	88.23	0.8831	91.14	0.9107	93.20	0.9298	90.86	0.9082	93.28	**0.9306**

Table 2.2: Performance comparision of the baseline models for activity recognition in PAMAP2, Opportunity and MMAR datasets.

CNN for each sensing modality whose output is then combined through a fully connected layer to identify the target class.

Evaluation Metrics We used the comparison of classification accuracy and mean F1-Score with the baseline models as our evaluation metrics. Classification accuracy is the number of correct predictions made as a ratio of all predictions made and the F1-score can be interpreted as a weighted average of precision and recall. The F1 scores can be calculated using the equation 2.3.

$$F1 - score = 2 * \frac{precision * recall}{precision + recall} \tag{2.3}$$

Table 2.2 compares the multimodal activity recognition performance of the different baseline models for the three datasets.

2.6.4 Training to obtain SenseHAR values

First, we obtain the SenseHAR values and the Application model using all the devices and sensor modalities by training them in the branched manner as explained in Section 2.4.3 for each dataset. We modified the K-max-pool layer of the sensor fusion model such that the sensor fusion model outputs SenseHAR values at a constant frequency of 25Hz (the smallest sampling frequency in the considered datasets). The training framework has one branch for every device and one branch for the combination of devices as shown in Figure 2.2 and all branches use the same Application model. Therefore, the framework for PAMAP2 dataset and MMAR dataset has 4 branches and Opportunity dataset has 6 branches.

Dataset	Branch	Sensors	Sampling Rate	Device Location	Accuracy	F1-score
PAMAP2	1	AGM	100	Wrist (W)	83.35	0.8347
	2	AGM	100	Ankle (A)	86.81	0.8664
	3	AGM	100	Chest (C)	87.24	0.8736
	4	AGM	100	W, A, C	**95.32**	**0.9508**
Opportunity (Gestures)	1	AGM	30	Back	65.14	0.5994
	2	AGM	30	RUA	67.32	0.6088
	3	AGM	30	RLA	63.81	0.5845
	4	AGM	30	LUA	66.74	0.6077
	5	AGM	30	LLA	62.05	0.5763
	6	AGM	30	Back, RUA, RLA, LUA, LLA	**74.22**	**0.6748**
MMAR	1	AGM	100	Pocket (P)	86.88	0.8658
	2	AG	50	Left Wrist (LW)	89.51	0.8955
	3	AG	25	Right Wrist (RW)	87.39	0.8747
	4	AGM	100,50,25	P, LW, RW	**93.13**	**0.9308**

Table 2.3: Performance of each branch from SenseHAR training framework for activity recognition in PAMAP2, Opportunity and MMAR datasets.The application model trained on the combination of SenseHAR values from all devices is able to achieve the state-of-the-art performance

After the training process, we get a sensor fusion model for every device that maps to SenseHAR and an Application model that works on the SenseHAR values. We obtain the accuracy and F1-scores on the test data and report the performance obtained by each branch in Table 2.3. Since the final branch in our framework corresponds to the Sensor fusion of all the available modalities we compare it with the performance of the baseline models shown in Table 2.2 and observe that our model achieves the state of the art performance of 95.32%, 74.22% and 93.13% on PAMAP2, Opportunity(gestures) and MMAR datasets respectively.

Dataset	Device Combination	F1-Scores					
		Ensemble	CNN	Conv-LSTM	MA-DNN	MA-CNN	SenseHAR-M
PAMAP2	Ankle, Chest	0.8765	0.8923	0.9218	0.8809	0.9210	**0.9326**
	Ankle, Wrist	0.8722	0.8735	0.8982	0.8677	0.8996	**0.9002**
	Wrist, Chest	0.8875	0.8931	0.9008	0.8845	0.9016	**0.9036**
Opportunity (Gestures)	RUA+LUA+Back	0.6226	0.6323	0.6533	0.6230	0.6568	**0.6630**
	RLA+LLA	0.5852	0.5952	**0.6197**	0.6018	0.6168	0.6190
	RUA+LLA	0.6148	0.6338	0.6392	0.6222	0.6420	**0.6554**
MMAR	Smartphone +Smartwatch	0.8966	0.8955	0.9120	0.8851	0.9202	**0.9308**
	Smartphone +Wristband	0.8842	0.8845	0.9095	0.8739	**0.9106**	0.9102
	Wristband +Smartwatch	0.8784	0.8811	0.9152	0.8812	0.9158	**0.9175**

Table 2.4: Performance comparison of the baseline models and SenseHAR-M (Model trained on SenseHAR values) for activity recognition considering different combination of devices in PAMAP2, Opportunity and MMAR datasets.

2.6.5 Different Combination of devices

We consider different scenarios where only a subset of the devices is available in each dataset. In these cases, we first obtain the SenseHAR values by using the sensor fusion model of the available devices. Then we take the arithmetic mean of the SenseHAR values from the available devices to combine their information and feed it to the trained Application model to get the activity predictions. We also train from scratch the baseline models for this different combination of devices to compare with our approach. Table 2.4 shows that our method of averaging the SenseHAR values to combine information and using the same Application model to make predictions performs equally well as training a new model from scratch for different combinations. This is of great significance because it eliminates the need to train a new model for every possible permutation of devices.

2.6.6 A new device with Different Sensors

In this case, we consider that the the user has a new device with only a subset of the sensors. In our experiments, the new devices are– 1. Chest sensor without gyroscope from PAMAP2

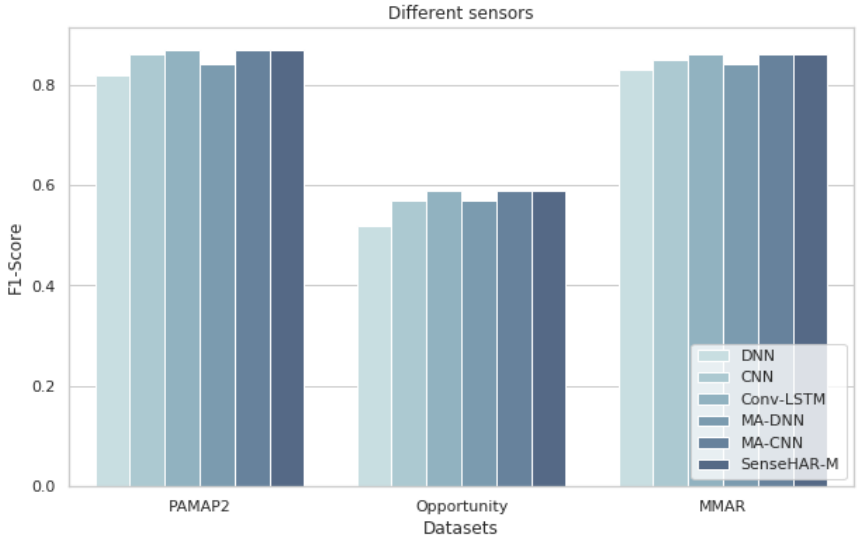


Figure 2.11: Performance comparison of the baseline models and SenseHAR-M (Model trained on SenseHAR values) for activity recognition in a new device with different sensors across datasets.

dataset, 2.RUA sensor without gyroscope and magnetometer from Opportunity dataset, 3. A smartphone without gyroscope and magnetometer from MMAR dataset. As seen from the table 2.3, these devices were not considered in the training phase. So now we initialize a new Sensor fusion model for each of the new devices which can take in the data from available sensors as input and train it using the SenseHAR values from a pre-trained device using the method explained in section 2.5.3. The Sensor Fusion model of the new devices learn to map to the same low dimensional latent space and the SenseHAR values obtained from this model is combined with the same previously trained Application model. We evaluate the performance on the test data and compare this performance to the performance of the baseline models (i.e.) the model that is trained from scratch on the available sensors and using the activity labels. We observe that even without using the true activity labels but training only on the SenseHAR values, this method is able to come very close to the performance of a baseline model trained on the true labels. There is a small performance drop of less than 1% as shown in Figure 2.11.

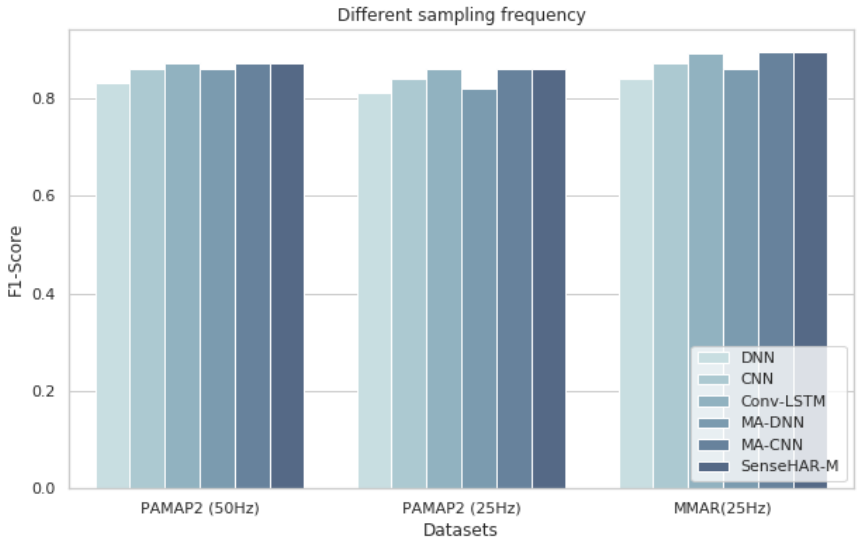


Figure 2.12: Performance comparison of the baseline models and SenseHAR-M (Model trained on SenseHAR values) for activity recognition in a new device with different sampling frequency across datasets.

2.6.7 A new device with a Different Sampling frequency

In this evaluation, we considered only the PAMAP2 and MMAR datasets since these datasets had frequencies higher than 25Hz. For this experiment, we considered two new devices from PAMAP2 dataset– 1. Wrist sensor that collects data at 50 Hz, 2. Ankle sensor that collects data at 25Hz and one new device from the MMAR dataset– 3. Smartwatch on the left wrist at 25Hz. The data for this experiment was obtained by downsampling the original data to from initial frequency to 50Hz and 25Hz respectively. We create a new sensor fusion model for each of these new devices such that they are able to support the new frequencies. This is done by adjusting the max-pooling layer of the sensor fusion model such that it maintains the same SenseHAR frequency of 25Hz. Then the sensor fusion models are trained in the same way as mentioned in section 2.5.3, combined with the Application model and evaluated on the test user data. We compare the performance with the baseline models trained from scratch on these new frequencies using the true activity labels. We find that our method is able to achieve similar performance as the baseline models though we trained it without using true activity labels. The performance metrics are shown in Figure 2.12.

2.6.8 A new hardware at a Different Location

To evaluate different locations, we used the Opportunity dataset since they had five different modalities based on locations (i.e) they had inertial sensors on five different locations on the body as seen in Figure 2.9. Instead of using all the five devices in the training framework as mentioned in section 2.6.4 we now use only four devices by leaving out one device. The left out device is then introduced as the new device in a different location. The experiment was done five times with the new device being in a different location each turn. A new sensor fusion model was initialized and calibrated for the new device. Figure 2.13 shows the performance of the new device when introduced in each of the five locations. We observe that, introducing an inertial sensor in a totally new location, calibrating a new sensor fusion model using the SenseHAR values from existing devices and combining it with the pre-trained Application model has an average performance drop of 5.2% and the worst performance drop of 6% (for the new device in 'Back' location) when compared to training a new baseline models from scratch using true activity labels. The drop in performance can be attributed to the bias in SenseHAR to the locations included in the training framework and can be mitigated by considering more devices located at different locations while training to obtain SenseHAR.

2.6.9 Different Activities

Usually, we initialize and train a new model on the sensor data for every new application and this is an extremely time and resource consuming process. So to avoid always training a model from scratch, we use the same Sensor fusion model for a particular device to obtain the SenseHAR values and then train an Application model which is a simple LSTM network on the SenseHAR values for the specific application. Training this simple LSTM model takes significantly less time than training a new model on raw sensor data since the SenseHAR already captures the high-level features corresponding to the activities. To test the generalizability of the SenseHAR values, we did it in two ways: (i) Train an Application model for a different application using the same SenseHAR values (ii) Test the performance

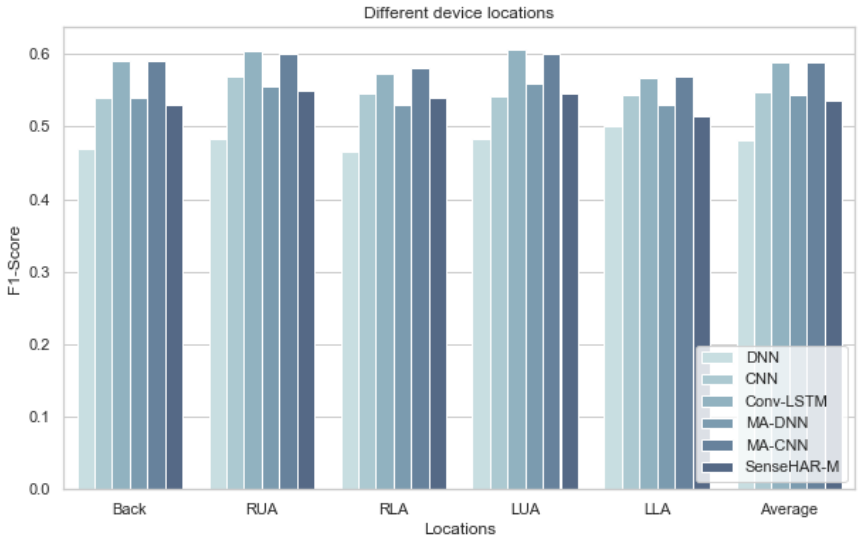


Figure 2.13: Performance comparison of the baseline models and SenseHAR-M (Model trained on SenseHAR values) for activity recognition in a new device at a new location in Opportunity dataset.

across datasets with similar hardware setup.

Different Application: Opportunity dataset has two different labels for each sensor stream- one for low-level gestures and another for higher level locomotion activity. In the training framework, while mapping to the shared latent space, we used the labels from low-level gestures and hence the Application model obtained will correspond to inferring the gestures. Now to infer the modes of locomotion, we train a new Application model on the SenseHAR data by using the same sensor fusion models. Table 2.5 shows the comparison of the performance metrics obtained by training the baseline models from the start for a different application and our proposed method.

Across Datasets: Data collection of MMAR dataset was similar to the PAMAP2 dataset which also had a wrist sensor worn on the participant's right wrist and hence, we obtained a Sensor fusion model from the PAMAP2 dataset for the wrist sensor sampled at 50Hz. We then used that Sensor fusion model on MMAR dataset to obtain the SenseHAR data for the smartwatch on the right wrist and trained a new Application model on the SenseHAR data for MMAR dataset. The performance of the new Application model was compared with baseline models from scratch on the training data. The performance comparison is shown

Dataset	Activites	F1-score	
		SenseHAR Model	Baseline
Opportunity (Locomotion)	Stand	0.9114	0.9048
	Sit	0.8346	0.8107
	Walk	0.9330	0.9600
	Lie	0.9466	0.9589
MMAR	Sit(idle)	0.9021	0.8992
	Sit(use Computer)	0.9045	0.9088
	Stand	0.9412	0.9488
	Walk	0.9227	0.9210
	Run	0.9574	0.9586

Table 2.5: Comparing individual activity recognition performance between the Application model trained on SenseHAR values and baseline model for Opportunity and MMAR datasets.

in Table 2.5. We found that the F1-score achieved by the Application model trained on SenseHAR values for each activity was similar to that of the baseline model. This shows that SenseHAR is able to capture the information of these activities performed by the participant. This is of importance because we are able to achieve similar performance as state of the art models in a hardware-agnostic way.

2.6.10 Training Time

We used GTX 1060 GPU to train our models. The initial training time to obtain the robust shared latent space representation is longer when compared to training an end-to-end model since the training framework involves optimization of the sum of the loss functions from multiple branches. It took 80 minutes for PAMAP2 and 50 minutes for Opportunity and MMAR datasets to obtain SenseHAR. But once we obtain the SenseHAR, as discussed in section 2.4 we do not have to train a new end-to-end model for different combination of devices since we can use the same application model and this saves time considerably.

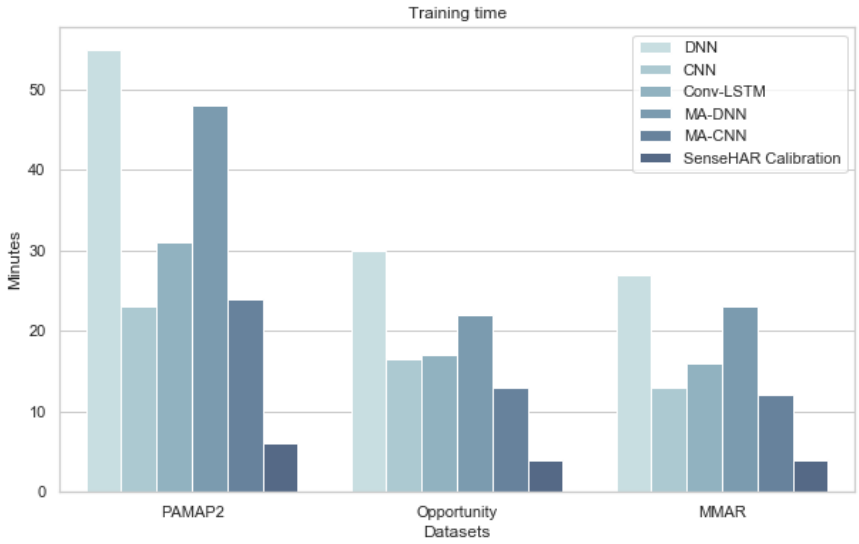


Figure 2.14: Comparison of time taken to train baseline models and to calibrate SenseHAR on a new device.

Additionally, the time taken to calibrate a new device to map to the same latent space is three to four times shorter than training a new end-to-end model as only a few convolutional layers in the sensor fusion model have to be trained. Figure 3.7 compares the time taken to calibrate the SenseHAR with the time taken to train the baseline models for a new device. It's important to note that we don't have to collect labeled data for calibrating SenseHAR which further saves the time and effort that would be required to label the data for every new device.

2.7 Related Work

In deep learning, before we use a neural network for a task (classification, regression), the usual architecture is to extract features through many layers (convolutional, recurrent, pooling, etc.). These layers map the inputs to the latent space on which the last classification layer is applied. Latent space has gained a lot of importance in Natural Language Processing(NLP) for word embeddings [GL14], [PSM14], in social network analysis for random graph models [HRH02] and in computer vision for applications like style transfer [ZPI17], data compression [HS06] and data interpolation [WZX16]. The work by Ngiam et al. [NKK11] shows how to learn a

shared representation between modalities and evaluate it on a unique task, where the classifier is trained with audio-only data but tested with video-only data and vice-versa.

For cross-domain activity recognition, the concept of transfer learning where certain parameters can be transferred while training a new model for a different application has been explored [MR16, CFK13] but this method still requires true activity labels to train the new model. Chen et al. [CWH19] introduce stratified transfer learning approach that uses pseudo labels from existing sensors instead of true labels. It performs intra-class knowledge transfer between domains iteratively to transform them into the same subspaces. Rokni et al. [RG18] introduce an autonomous training method where, pseudo labels obtained from existing sensors are concatenated with features from the new sensor and then clustered to improve accuracy. But these approaches only work on the traditional method of extracting features from the raw sensor values. Xing et al. [XSB18] show that the shared latent space representation exists for time series sensory data, and it can help transfer knowledge from ambiance edge devices to wearable edge devices and vice versa. Radu et al.'s deep learning based modality specific architecture [RTB18] does sensor fusion in latent space. But this work does not take into consideration the heterogeneities in the devices and sensors and hence, a new end-to-end model has to be trained for each device and application. Also, these models assume that all the modalities are present at any given time which is generally not the case in real life implementations. The work by Khan et al. [KRM18] assumes that the distribution of weights and biases in the convolutional layers remains largely unchanged across different activities, and thus automatically adapts and learns the model across different domains with minimal labeled data. This approach does not address the problem of a new device having a different hardware configuration than the existing device. The conventional procedure to take care of the difference in sampling rates of the devices is to interpolate or resample the collected data to the required frequency. The two common methods of interpolation used for HAR are linear [CSC13] and cubic spline [TIH07]. This means additional preprocessing has to be done before feeding it to the HAR models. Stisen et al. [SBB15] propose a clustering-based approach as a mitigation technique to improve HAR performance in the presence of heterogeneities but their work is limited to devices with a single modality(accelerometer).

One of the early efforts in creating a virtual sensor from sensor fusion was the work on virtual gyroscope [CXQ08] which proposed the theory of gyro-free IMU. The virtual gyroscope with multigyroscope and accelerometer array (MGAA)configuration merges the outputs of multi-gyroscopes and specifically placed accelerometers through a Kalman filter. The work by LiKamWa et al. [LLL13] proposed Moodscope, a virtual mood sensor for a smartphone that measures an important mental state of the user based on the interactions with the smartphone. These papers relied on traditional signal processing and decision-based algorithms and hence do not generalize well for different devices and different users.

2.8 Discussion & Future Work

Training Process: For each dataset, the sensor sequences were divided into window size of five seconds with one-second overlap. When the number of sensor modalities changes, we change the input shape of the first convolutional layer in the Sensor fusion model to handle this difference. When the sensors had different sampling frequencies, we adjusted the K-max-pool layers to support the different frequencies. In our implementation, the sensor fusion model had one or two max-pool layers with a pool size of two if the input sensor data was sampled at 50Hz or 100Hz respectively. When the sampling frequency was 30 Hz, we adjusted the K-max-pool layer such that it returns the maximum five values from a pool size of six. Thus when high frequency sensor values are fed into the sensor fusion model, the output, which is the SenseHAR values, will always have a constant frequency of 25Hz. Though the frequency of the SenseHAR is less than the input raw sensor frequencies, the relevant high-frequency features from the sensors are captured by the convolutional and pooling layers of the sensor fusion model, and is rich in information. In addition to the convolutional, K-max-pool, LSTM and dense layers, we also used batch normalization [IS15] and dropout layers [SHK14] in our models. The use of batch normalization reduces the impacts of earlier layers by keeping the mean and variance fixed, which makes the layers independent with each other and helps the model to train faster. We used dropout layers as the method of regularization to prevent over-fitting of our models. In the dropout layer, the

neurons are randomly dropped out of the network with a certain probability during training. This makes the other neurons to step in and handle the representation required to make predictions for the missing neurons resulting in multiple independent internal representations being learned by the network. We used Adam optimizer [KB14], an extension of the classical stochastic gradient descent procedure, which maintains an adaptive, per-parameter learning rate which helps the network to converge faster.

Pre-Processing: Since our method works on raw sensor streams it requires no pre-processing before the training phase. Hence, we don't require domain-specific expertise to do feature engineering. This saves the effort required to do identify the important features and also the time required to compute those features. We used a window size of five seconds as inputs to our model so that we have sufficient information from each sensor to capture the high-level features and map them to a shared latent space.

Limitations and Future work: To make sure that the SenseHAR values are not biased to the applications it is trained on, we need a significantly large dataset with diverse activities. In our experiments, since the maximum sampling frequency of the devices we considered was 100Hz, the performance of SenseHAR was evaluated only on activities that last for a few seconds. We did not consider any minute fine-grained activities that occur in the millisecond scale (lasts for only a few milliseconds) as it requires data sampled at high frequencies to capture the fine grained information. In the current method, our sensor fusion model does not work for devices having sensors with sampling frequencies less than 25Hz. So in the future, we aim to address this issue by trying different techniques like zero padding to the input sequence or adding de-convolutional layers to the sensor fusion model to support any arbitrary sampling rates. Also, we used the arithmetic mean for combining information because we wanted the shared latent space obtained from our training framework to be independent of the type of device or the device configuration or location and only contain information related to the activities and hence we gave equal weights to all the devices. In future we plan to incorporate techniques like adaptive weighted averaging and explore different normalization techniques at the sensor fusion step, which might help in improving the performance.

2.9 Conclusion

In this paper, we explore the different challenges and heterogeneities in smartphones and wearables for HAR and construct SenseHAR– a virtual activity sensor that is robust to the variations and availability of the inertial sensors using deep learning. SenseHAR enables the inferring of human activities even when the set of available devices varies over time by mapping the information from the available devices to a shared low-dimensional latent space. We explain the method to calibrate the SenseHAR for a new device in the absence of labeled data with the help of an existing calibrated device. Finally, we do a comprehensive evaluation of SenseHAR and compare its performance with the state-of-the-art multimodal architectures for various device combinations and configurations. Our results indicate that SenseHAR generalizes well for a different set of devices, different sampling frequency, different sensors and for different applications. It shows a small degradation in performance when calibrating for a device introduced in a new location on the body which wasn't included in the training framework. This drop in performance can be minimized if we include a wide array of devices located at different locations on the body in the training framework. Our virtual activity sensor - SenseHAR decouples hardware from software and will help application developers to not worry about the hardware specifications of the device while creating activity recognition models.

CHAPTER 3

Combining Individual and Joint Networking Behavior for Intelligent IoT Analytics

The IoT vision of a trillion connected devices over the next decade requires reliable end-to-end connectivity and automated device management platforms. While we have seen successful efforts for maintaining small IoT testbeds, there are multiple challenges for the efficient management of large-scale device deployments. With Industrial IoT, incorporating millions of devices, traditional management methods do not scale well. In this work, we address these challenges by designing a set of novel machine learning techniques, which form a foundation of a new tool, *IoTelligent*, for IoT device management, using traffic characteristics obtained at the network level. The design of our tool is driven by the analysis of 1-year long networking data, collected from 350 companies with IoT deployments. The exploratory analysis of this data reveals that IoT environments follow the famous Pareto principle, such as: (i) 10% of the companies in the dataset contribute to 90% of the entire traffic; (ii) 7% of all the companies in the set own 90% of all the devices. We designed and evaluated CNN, LSTM, and Convolutional LSTM models for demand forecasting, with a conclusion of the Convolutional LSTM model being the best. However, maintaining and updating individual company models is expensive. In this work, we design a novel, scalable approach, where a general demand forecasting model is built using the combined data of all the companies with a normalization factor. Moreover, we introduce a novel technique for device management, based on autoencoders. They automatically extract relevant device features to identify device groups with similar behavior to flag anomalous devices.

3.1 Introduction

The high-tech industry expects a trillion new IoT devices will be produced between now and 2035 [mas16, him18, 1T20]. These devices could range from simple sensors in everyday objects to complex devices, defined by the industrial and manufacturing processes. The Internet of Things ecosystem should include the necessary components that enable businesses, governments, and consumers to seamlessly connect to their IoT devices. This vision requires reliable end-to-end connectivity and device management platform, which makes it easier for device owners to access their IoT data and exploiting the opportunity to derive real business value from this data. The benefits of leveraging this data are greater business efficiencies, faster time to market, cost savings, and new revenue streams. Embracing these benefits ultimately comes down to ensuring the data is secure and readily accessible for meaningful insights.

The Arm Mbed IoT Device Management Platform [pel20a] addresses these requirements by enabling organizations to securely develop, provision and manage connected devices at scale and by enabling the connectivity management [pel20b] of every device regardless of its location or network type. The designed platform supports the physical connectivity across all major wireless protocols (such as cellular, LoRa, Satellite, etc.) that can be managed through a single user interface. Seamlessly connecting all IoT devices is important in ensuring their data is accessible at the appropriate time and cost across any use case. While we could see successful examples of deploying and maintaining small IoT testbeds, there are multiple challenges in designing an efficient management platform for large-scale device deployments. The operators of IoT environments may not be fully aware of their IoT assets, let alone whether each IoT device is functioning and connected properly, and whether enough networking resources and bandwidth allocated to support the performance objectives of their IoT networks. With the IoT devices being projected to scale to billions, the traditional (customized or manual) methods of device and IoT networks management do not scale to meet the required performance objectives.

In this work, we aim to address these challenges by designing a set of novel machine

learning techniques, which form a foundation of a new tool, *IoTelligent* [JCL20], for IoT networks and device management, using traffic characteristics obtained at the network level. One of the main objectives of *IoTelligent* is to build effective demand forecasting methods for owners of IoT ecosystems to manage trends, predict performance, and detect failures. The insights and prediction results of the tool will be of interest to the operators of IoT environments.

For designing the tool and appropriate techniques, we utilize the unique set of real (anonymized) data, which were provided to us by our business partners. This dataset represents 1-year of networking data collected from 350 companies with IoT deployments, utilizing the Arm Mbed IoT Device Management Platform. The exploratory analysis of the underlying dataset reveals a set of interesting insights into the nature of such IoT deployments. It shows that the IoT environments exhibit properties similar to the earlier studied web and media sites [AW96, Kim01, CG04, TFC07] and could be described by famous Pareto principle [wik], when the data distribution follows the power law [pow]. The Pareto principle (also known as the 80/20 rule or the "law of the vital few") states that for many events or data distributions roughly 80% of the effects come from 20% of the causes. For example, in the earlier web sites, 20% of the web pages were responsible for 80% of all the users accesses [AW96]. The later, popular web sites follow a slightly different proportion rule: they often are described by 90/10 or 90/5 distributions, i.e., 90% of all the user accesses are targeting a small subset of popular web pages, which represent 5% or 10% of the entire web pages set.

The interesting findings from the studied IoT networking dataset can be summarized as follows:

- 10% of the companies in the dataset contribute to 90% of the entire traffic;
- 7% of all the companies in the dataset own 90% of all the devices.

IoTelligent tool applies machine learning techniques to forecast the companies' traffic demands over time, visualize traffic trends, identify and cluster devices, detect device anomalies and failures. We designed and evaluated CNN, LSTM, and Convolutional LSTM models for

demand forecasting, with a conclusion of the Convolutional LSTM model being the best. To avoid maintaining and upgrading tens (or hundreds) of models (a different model per company), we designed and implemented a novel, scalable approach, where a global demand forecasting model is built using the combined data of all the companies. The accuracy of the designed approach is further improved by normalizing the "contribution" of individual company data in the combined global dataset. To solve the scalability issues with managing the millions of devices, we designed and evaluated a novel technique based on: (i) autoencoders, which extract the relevant features automatically from the network traffic stream; (ii) DBSCAN clustering to identify the group of devices that exhibit similar behavior, to flag anomalous devices. The designed management tool paves the way the industry can monitor their IoT assets for presence, functionality, and behavior at scale without the need to develop device-specific models.

3.2 Dataset and the Exploratory Data Analysis

The network traffic data was collected from more than 350 companies for a total duration of one year. The traffic data is binned using 15 minute time window, used for billing purposes.

- Unix timestamp;
- Anonymous company ids;
- Anonymous device ids per company;
- The direction of the network traffic (to and from the device);
- Number of bytes transmitted in the 15 minute interval;
- Number of packets transmitted in the 15 minute interval.

Preliminary analysis was done to find the most impactful and well-established companies. We found that the companies' data that represent two essential metrics, such as the networking traffic amount and number of deployed IoT devices, both follow the Pareto law. The main findings from the studied IoT networking dataset can be summarized as follows:

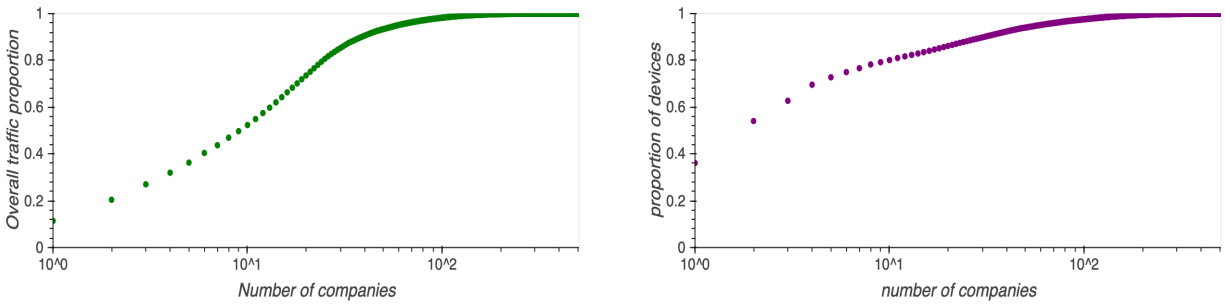


Figure 3.1: (left) CDF of networking traffic; (right) CDF of devices.

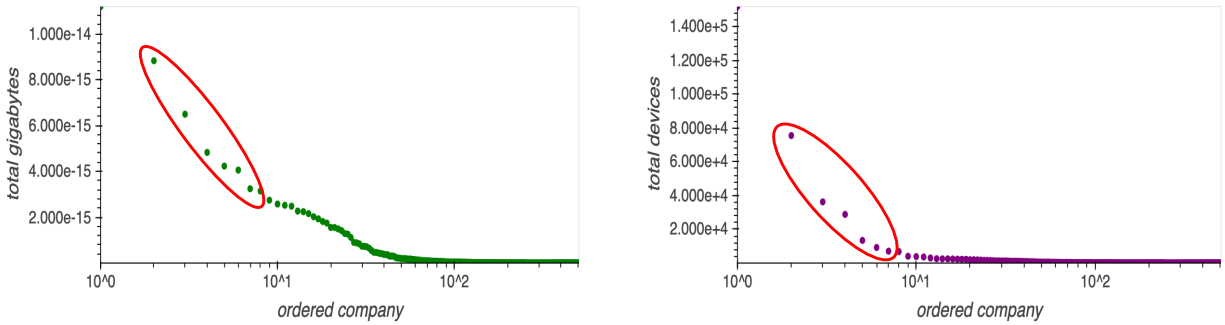


Figure 3.2: (left) Networking traffic per company; (right) Number of devices per company.

- 10% of the companies in the dataset contribute to 90% of the entire traffic;
- 7% of all the companies in the dataset own 90% of all the devices.

Figure 3.1 shows on the left side the logscale graph of CDF (Cumulative Distribution Function) of the traffic (where one can see that 10% of the companies in the dataset contribute to 90% of the entire traffic) and the CDF of the devices per company distribution (where one can see that 7% of all the companies in the dataset own 90% of all the devices). Also, it is quite interesting to note how significant and sizable the contributions are of the first 5-10 companies on those graphs: both for the networking traffic volume and the number of overall devices.

Another interesting observation was that companies with highest number of devices did not correspond to companies with maximum amount of traffic, and vice versa, the high volume traffic companies did not have a lot of devices. This makes sense, for example, a difference in the outputs of hundreds of simple sensors and a single recording camera. Among some other insights into special properties of many IoT environments (at the networking

level) we observe the pronounced diurnal and weekly patterns, and changes in the traffic patterns around some seasonal events and holidays. It could be explained by the fact that many IoT environments are related to human and business activities.

3.3 Demand Forecasting

The demand forecasting problem is formulated in the following way. Given a recent month's traffic pattern for a company, what is the expected traffic for this company a week ahead? This problem requires that a predictive model forecasts the total number of bytes for each hour over the next seven days. Technically, this framing of the problem is referred to as a multi-step time series forecasting problem, given the multiple forecast steps. Choosing the right time granularity for (i) making the prediction and (ii) data used in the model, is another important decision for this type of a problem.

We found that a reasonable trade-off would be to use 1 hour time granularity. This eliminates the small noises in traffic and also ensures that we have a sufficient data to train our models on.

3.3.1 Modeling Approach

Based on our exploratory data analysis, we select 33 companies with largest traffic and 5 companies with largest number of devices. These companies are responsible for 90% of the networking traffic volume and 90% of IoT devices. Therefore, by designing and evaluating the modeling approach for these companies, we could efficiently cover the demand forecasting for 90% of the traffic volume and assessing the monitoring solution for 90% of devices.

The specific goal is to predict the company traffic for a next week given the previous three weeks of traffic data in an hourly time granularity. We use a deep learning based approach for demand forecasting, because deep learning methods are robust to noise, highly scalable, and generalizable. We have considered three different deep learning architectures for demand forecasting: CNN, LSTM, and Convolutional LSTM in order to compare their outcome and accuracy.

Convolutional Neural Network (CNN) [KSH12]: It is a biologically inspired variant of a fully connected layer, which is designed to use minimal amounts of preprocessing. CNNs are made of Convolutional layers that exploit spatially-local correlation by enforcing a local connectivity pattern between neurons of adjacent layers. The main operations in Convolution layers are Convolution, Activation (ReLU), Batch normalization, and Pooling or Sub-Sampling. The CNN architecture, used in our experiments, has 4 main layers. The first three layers are one-dimensional convolutional layers, each with 64 filters and relu activation function, that operate over the 1D traffic sequence. Each convolutional layer is followed by a max-pooling layer of size 2, whose job is to distill the output of the convolutional layer to the most salient elements. A flatten layer is used after the convolutional layers to reduce the feature maps to a single one-dimensional vector. The final layer is a dense fully connected layer with 168 neurons (24 hours x 7 days) with linear activation and that produces the forecast by interpreting the features extracted by the convolutional part of the model.

Long Short Term Memory (LSTM) [GSC99, HS97]: It is a type of Recurrent Neural Network (RNN), which takes current inputs and remembers what it has perceived previously in time. An LSTM layer has a chain-like structure of repeating units and each unit is composed of a cell, an input gate, an output gate, and a forget gate, working together. It is well-suited to classify, process, and predict time series with time lags of unknown size and duration between important events. Because LSTMs can remember values over arbitrary intervals, they usually have an advantage over alternative RNNs, Hidden Markov models, and other sequence learning methods in numerous applications. The model architecture, used in our experiments, consists of two stacked LSTM layers, each with 32 LSTM cells, followed by a dense layer with 168 neurons to generate the forecast.

Convolutional LSTM [XCW15]: Convolutional LSTM is a hybrid deep learning architecture that consists of both convolutional and LSTM layers. The first two layers are the one-dimensional Convolutional layers that help in capturing the high-level features from the input sequence of traffic data. Each convolutional layer is followed by a max-pooling layer to reduce the sequence length. They are followed by two LSTM layers, that help in tracking the temporal information from the sequential features, captured by the convolutional layers. The

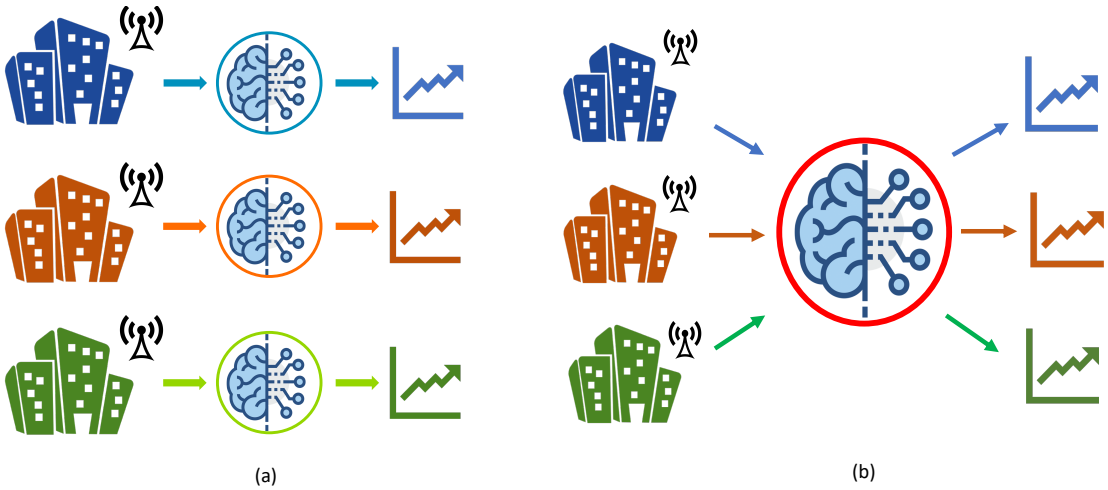


Figure 3.3: (a) Each company has its own prediction model, (b) Using one model for all the companies, trained on the combined dataset.

final layer is a dense fully connected layer, that gives a forecasting output.

We use batch-normalization and dropout layers in all our models. To evaluate the prediction accuracy of the designed models, we compare the predicted value X_n^{pred} with the true, measured value X_n using following error metrics:

Mean Absolute Error (MAE):

$$MAE = \frac{1}{N}\Sigma_{n=1}^{N}|X_n - X_n^{pred}|$$

Mean Squared Error (MSE):

$$MSE = \frac{1}{N}\Sigma_{n=1}^{N}(X_n - X_n^{pred})^2$$

3.3.2 Individual Model Per Company

This is the naive approach where each company has it's own demand forecasting model, that is, the model for each company is trained by using only the data from that particular company as shown in Figure 3.3 (a).

So, for each company, we trained three models with the architectures described above (i.e., CNN, LSTM, and Convolutional LSTM). Figure 3.4 (a) presents the detailed parameters of the designed Convolutional LSTM, while Figure 3.4 (b) reflects the relative performance of three

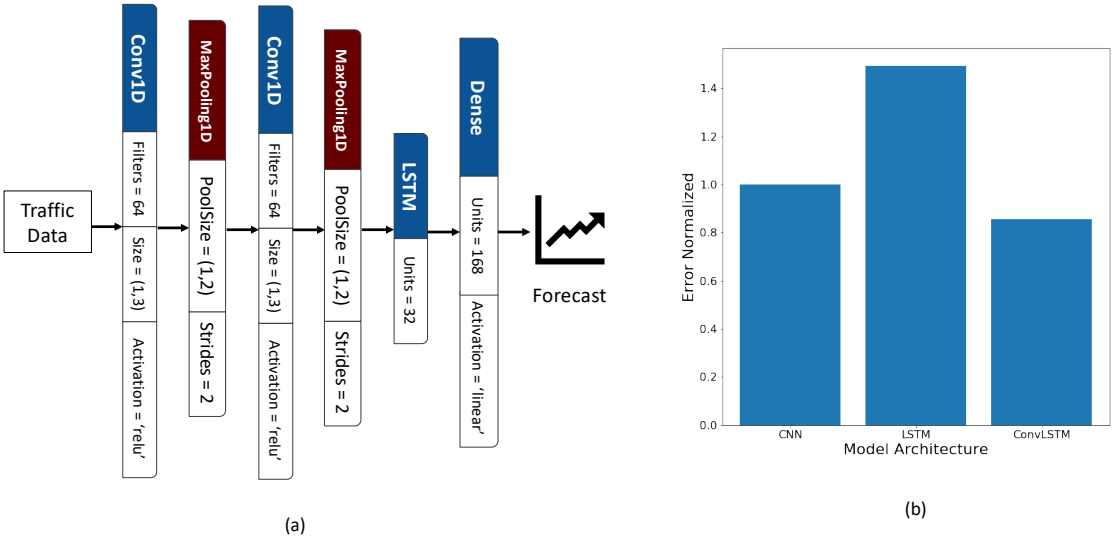


Figure 3.4: (a) Model Architecture of Convolutional LSTM Model; (b) Comparing performance of the three architectures: Convolutional LSTM achieves best performance.

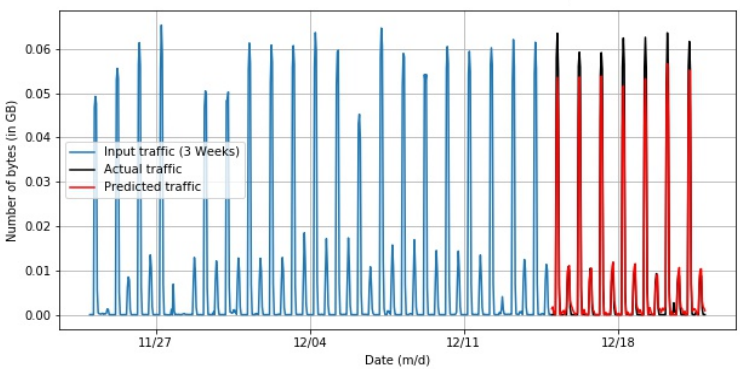


Figure 3.5: Company A: the 4th week demand forecast based on data from the previous 3 weeks.

different architectures (with the MAE error metrics). We found that for both error metrics the Convolutional LSTM model performs better than the other two architectures. When comparing architectures' accuracy by using MAE and MSE, we can see that Convolutional LSTM outperforms CNN by **16%** and **23%** respectively, and outperforms LSTM by **43%** and **36%** respectively. Therefore, only **Convolutional LSTM** architecture is considered for the rest of the paper. Finally, Figure 3.5 shows an example of company A (in the studied dataset): its measured networking traffic over time and the forecasting results with the Convolutional LSTM model.

Building an individual model per each company has a benefit that this approach is simple to implement. But this method has significant drawbacks. First, the model easily overfits on the training data since it's trained only using limited data from a particular company resulting in poor forecasting performance. Secondly, it is not scalable as the number of required forecasting models is directly proportional to the number of companies. The service provider has to deal with the models' maintenance, their upgrades, and retraining (with new data) over time. Therefore, in the next Section 3.3.3, we aim to explore a different approach, which enables a service provider to use all the collected, global data for building a single (global) model, while using it for individual company demand forecasting. Only Convolutional LSTM architecture is considered in the remaining of the paper (since as shown, it supports the best performance).

3.3.3 One Model for All Companies - Without Normalization

In this approach, we train a single Convolutional LSTM model for demand forecasting by using data from all the companies. The networking traffic data from all the companies were combined. The data from January to October were used for training the model, and the data from November and December were used as the test set.

This method is highly scalable since it trains and utilizes a single model for demand forecasting of all companies. While this approach is very attractive and logical, it did not always produce good forecasting results. Figure 3.6 shows the forecasting made by this global

model for Company A (with this company we are already familiar from Figure 3.5). As we can see in Figure 3.6, the model fails to capture a well-established traffic pattern.

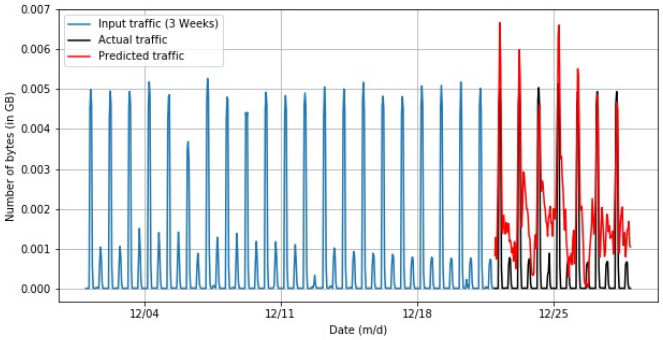


Figure 3.6: Demand forecasting using the Global model trained on data without normalization.

One of the explanations of the observed issue is that this company's traffic constitutes a very small fraction compared to the other companies in the combined dataset. So, the globally trained model has "learned" the traffic patterns of larger companies in the set, while has "downplayed" the traffic patterns of smaller companies. The reason the model fails to capture a well-established traffic pattern for companies with less traffic is because, the traffic prediction loss in terms of absolute value is still small. However, it is not a desirable outcome as we would like our model to capture the traffic pattern even for companies with low traffic.

3.3.4 One Model for All Companies - With Normalization

This method aims to address the issues of the previous two approaches. In this method, the data from each company is normalized, that is, all the data subsets are scaled so that they lie within the same range. We use the min-max scaling approach to normalize the data subsets so that the values of the data for all companies lie between 0 and 1. Equation 3.1 shows the formula used for min-max scaling, where 'i' refers to the 'i'th company.

$$X^i_{norm} = \frac{X^i - X^i_{min}}{X^i_{max} - X^i_{min}} \tag{3.1}$$

Then a single deep learning model for forecasting is trained using the normalized data of all companies. The predicted demand (forecast) is then re-scaled using Equation 3.2 to the original scale.

$$X^i = X^i_{norm} * (X^i_{max} - X^i_{min}) + X^i_{min} \tag{3.2}$$

This method of training the global model gives equal importance to the data from all companies and treats them fairly. The designed model does not over-fit and is generalizable since it is trained on the data from multiple companies. Figure 3.7 graphically reflects the process of the global model creation with normalized data from different companies.

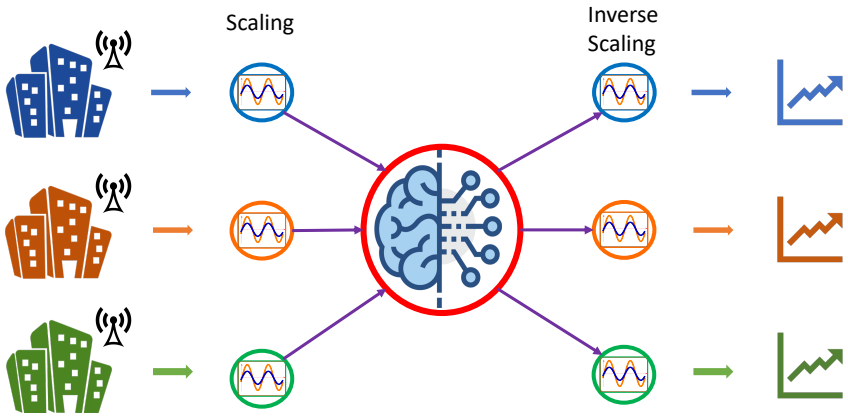


Figure 3.7: One global prediction model is trained by using the normalized data from all the companies.

Figure 3.8 shows that the designed forecasting model can capture well the patterns of companies with low traffic volume (such as Company A).

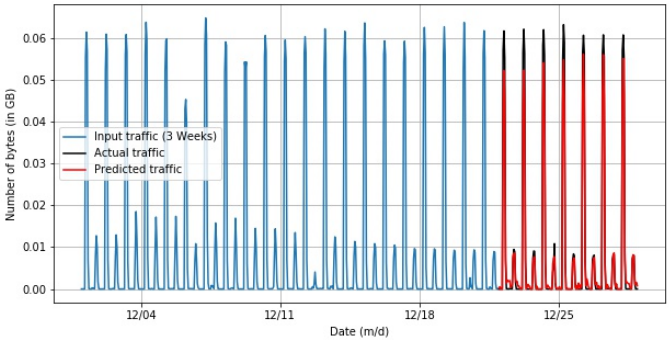


Figure 3.8: Demand forecasting using the Global model trained on data with normalization.

3.4 Introducing Uncertainty to Forecasting Models

In the previous section, we designed a single global model with normalization, that can be used to forecast for multiple companies. But demand forecasting is a field, where an element of uncertainty exists in all the predictions, and therefore, representing model uncertainty is of crucial importance. The standard deep learning tools for forecasting do not capture model

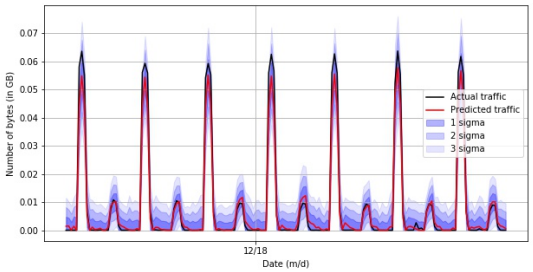


Figure 3.9: Demand forecasting with uncertainty for Global model trained on data with normalization.

uncertainty. Gal et. al [GG16] propose a simple approach to quantify the neural network uncertainty, which shows that the use of dropout in neural networks can be interpreted as a Bayesian approximation of a Gaussian process - a well known probabilistic model. Dropout is used in many models in deep learning as a way to avoid over-fitting, and it acts as a regularizer. However, by leaving it “on" during the prediction, we end up with the equivalent of an ensemble of subnetworks, within our single larger network, that have slightly different views of the data. If we create a set of T predictions from our model, we can use the mean and variance of these predictions to estimate the prediction set uncertainty. Figure 3.9 shows the forecast with uncertainty for Company A, using the global model with normalization.

To evaluate the quality of forecast based on uncertainty, we introduce Prediction Interval Coverage Probability (PICP) metric.

3.4.1 Prediction Interval Coverage Probability (PICP)

PICP tells us the percentage of time an interval contains the actual value of the prediction. Equations 3-5 show the calculation of PICP metric, where l is the lower bound, u is the upper bound, x_i is the value at timestep i, $\hat{y}$ is the mean of the predicted distribution, z is the number of standard deviations from the Gaussian distribution, (e.g., 1.96 for a 95% interval), and σ is the standard deviation of the predicted distribution.

$$l(x_i) = \hat{y}_i - z * \sigma_i \tag{3.3}$$

$$u(x_i) = \hat{y}_i + z * \sigma_i \tag{3.4}$$

$$PICP_{l(x),u(x)} = \frac{1}{N}\sum_{i=1}^{N} h_i, \quad where\ h_i = \begin{cases} 1, & if\ l(x_i) \leq y_i \leq u(x_i) \\ 0, & otherwise \end{cases} \tag{3.5}$$

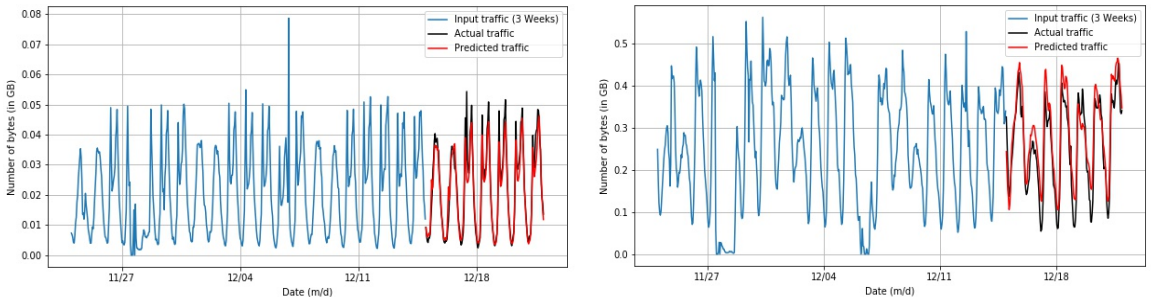


Figure 3.10: Demand forecasting with Global model trained on data with normalization.

3.4.2 Evaluating Forecast with Uncertainty

We evaluate the overall performance of our global forecast model, introduced in Section 3.3.4, based on the PICP metric described above. The forecasting is done 100 times for each company with a dropout probability of 0.2, and then the mean and standard deviations are obtained for each company. Figure 3.10 shows the global model's forecast for the third week of December for two different companies: Company B and Company C. As we can see from the plot, the model captures the traffic pattern, but still, the predicted values show some deviations from the actual values. This results in some errors when using the traditional error metrics discussed in Section 3.3.1, though the model is performing very well. Therefore, introducing uncertainty helps the model to generate a reasonable forecast distribution. Figure 3.11 shows the forecast with uncertainty, where the different shades of blue indicate the uncertainty interval, obtained for different values of uncertainty multipliers. As we can see from the plot, the single global forecasting model can capture well the general traffic trends across multiple companies. Figure 3.12 shows the mean PICP calculated across all the companies for the different uncertainty multipliers.

We find that on an average 50% of the forecast values lie within the predicted interval with one standard deviation, 74% for two standard deviations and 85% for three standard deviations. The forecast samples which lied outside the predicted interval were mostly due to the fact that the months of November and December had lots of holidays and hence those days did not follow the captured traffic pattern.

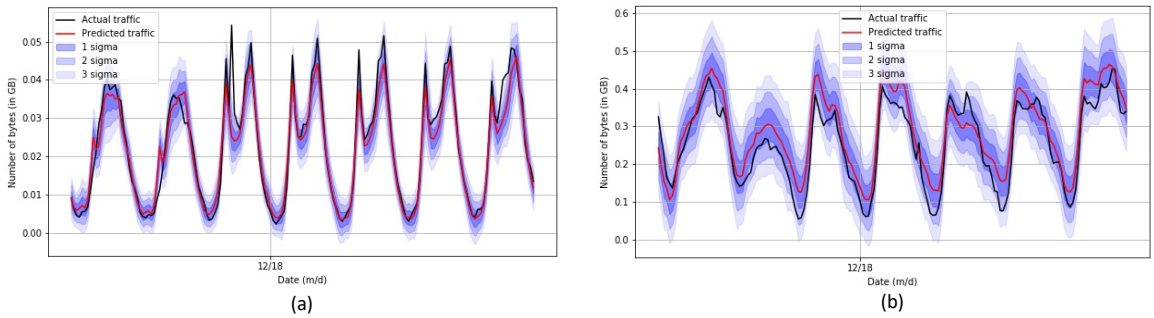


Figure 3.11: Demand forecasting with Uncertainty using the Global model trained on data with normalization.

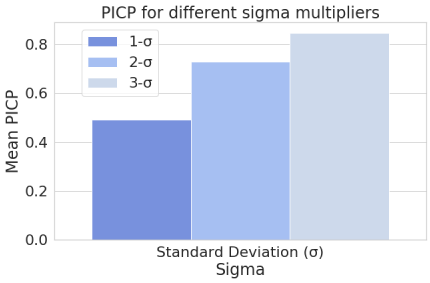


Figure 3.12: Mean PICP for different values of sigma multiplier.

3.5 Device Monitoring and Diagnostics

Once an Internet of Things (IoT) ecosystem is installed, it does not follow a “fire and forget” scenario. There will be unforeseen operational issues, some devices will fail and/or would need to be either repaired or replaced. Each time this happens, the company is on the mission to minimize the downtime and ensure that its devices function properly to protect their revenue stream. However, to address the issues of failed or misbehaving devices, we need to identify such devices in the first place. Therefore, the ability to monitor the device’s health and being able to detect, when something is amiss, such as higher-than-normal network traffic or “unusual" device behavior, it is essential to proactively identify and diagnose potential bugs/issues. Again, large-scale IoT deployments is a critical and challenging issue. When there are thousands of devices in the IoT ecosystem, it becomes extremely difficult to efficiently manage these devices as it is practically impossible to monitor each device individually. So, we need an efficient way to analyze the observed device behaviors and identify devices that

show an anomalous (“out of usual norm") behavior.

Anomalous or failed devices can be categorized into two types:

1. The devices that behave significantly different from the other devices;
2. The devices whose observed behavior suddenly changes from its “normal" behavior over time.

The following Section describes the designed technique to accomplish the device monitoring and diagnostic via device categorization over time.

3.5.1 Cluster Devices based on their Traffic Patterns and Identify Anomalous Devices

When there are thousands of devices in a given IOT Ecosytem, there usually exist multiple devices of the same type or having similar behavior. We identify these groups of devices in an unsupervised manner based on their network traffic pattern over a given month. Figure 3.13 shows an overview of the proposed method and its steps to obtain the groups of "similar" devices:

- The monthly network traffic from the thousands of IoT devices are passed through an autoencoder to extract features in the latent space in an unsupervised manner.
- Then we use a density-based clustering algorithm, DBSCAN, on the latent space to identify the groups of similar devices. The objective is to learn what normal data points looks like and then use that to detect abnormal instances. Any instance that has a low affinity to all the clusters is likely to be an anomaly.

3.5.1.1 Autoencoder [MMC11]

It is a neural network capable of learning dense representations of the input data, called latent space representations, in an unsupervised manner. The latent space has low dimensions

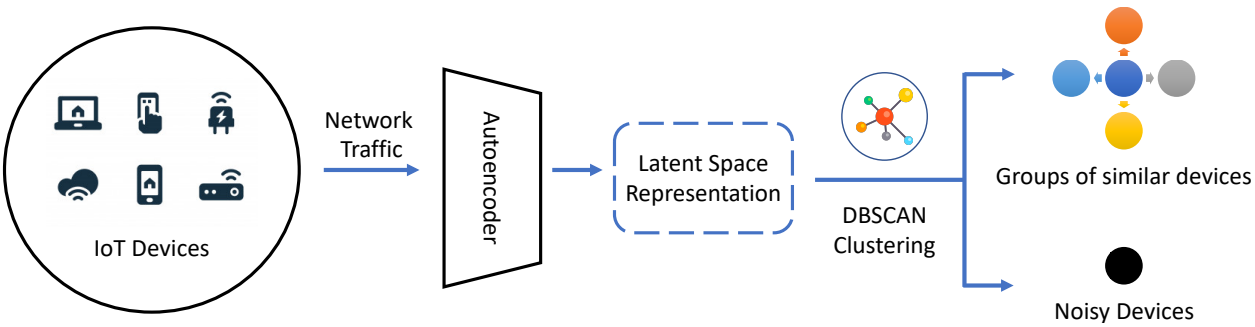


Figure 3.13: The pipeline for clustering similar groups of devices and detecting noisy devices based on their traffic patterns.

which helps in visualization and dimensionality reduction. An autoencoder has two parts: an encoder network that encodes the input values x, using an encoder function f, and, a decoder network that decodes the encoded values f(x), using a decoder function g, to create output values identical to the input values. Autoencoder's objective is to minimize reconstruction error between the input and output. This helps autoencoders to capture the important features and patterns present in the data in a low dimensional space. When a representation allows a good reconstruction of its input, then it has retained much of the information present in the input. In our experiment, an autoencoder is trained using the monthly traffic data from the IoT devices which captures the important features or the encoding of the devices in the latent space.

Architecture of the Autoencoder: We use a stacked autoencoder in our experiment with two fully connected hidden layers each in the encoder and the decoder. The central bottle neck layer was a fully connected layer with just three neurons which helps in reducing the dimensions. We used mean squared error as the reconstruction loss function.

3.5.1.2 DBSCAN [EKS96]

(Density-Based Spatial Clustering of Applications with Noise), is a density-based clustering algorithm that captures the insight that clusters are dense groups of points. If a particular point belongs to a cluster, it should be near to lots of other points in that cluster. The algorithm works in the following order: First, we choose two parameters, a positive number, epsilon and a natural number, minPoints. We then begin by picking an arbitrary point in

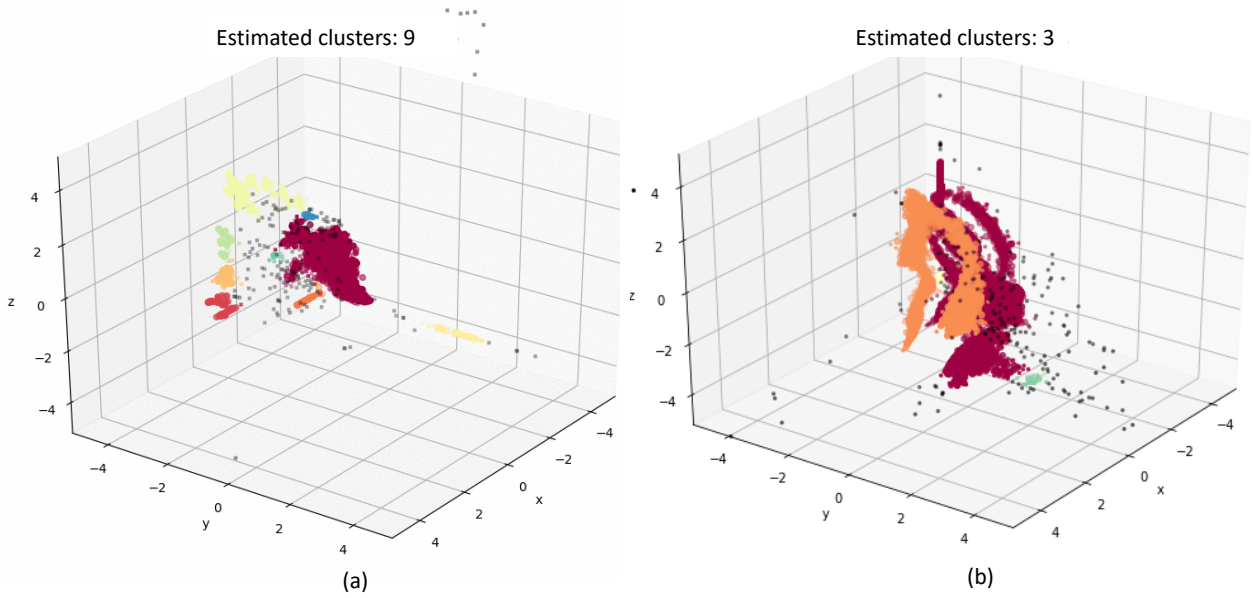


Figure 3.14: Number of Clusters per company when visualized in the latent space. The black points represent the anomalous devices

our dataset. If there are more than minPoints points within a distance of epsilon from that point, (including the original point itself), we consider all of them to be part of a "cluster". We then expand that cluster by checking all of the new points and seeing if they too have more than minPoints points within a distance of epsilon, growing the cluster recursively if so. Eventually, we run out of points to add to the cluster. We then pick a new arbitrary point and repeat the process. Now, it's entirely possible that a point we pick has fewer than minPoints points in its epsilon ball, and is also not a part of any other cluster. If that is the case, it's considered a "noise point" not belonging to any cluster and we mark that as an anomaly.

Figure 3.14 shows the latent space and the clusters obtained for Company A(left) and Company B(right). Companies A and B had more than 30000 devices each, installed in their IoT ecosystems and they had three and nine unique types of devices respectively. Based on the devices' traffic patterns observed over the period of a month, the autoencoder mapped the devices of the same type close to each other while the devices of different types were mapped far apart from each other in the latent space. When DBSCAN clustering was applied in the latent space, we observed that the number of distinct clusters formed was exactly the same as the corresponding number of device types per company. The devices which didn't

fall in these well formed clusters because of their different traffic patterns were marked as anomalies and are represented by the black points.

3.6 Related Work

Demand forecasting has been broadly studied due to the problem importance and its significance for utility companies.Statistical methods use historical data to make the forecast as a function of most significant variables. The detailed survey on regression analysis for the prediction of residential energy consumption is offered in [ACA19] where the authors believe that among statistical models, linear regression analysis has shown promising results because of satisfactory accuracy and simpler implementation compared to other methods. In many cases, the choice of the framework and the modeling efforts are driven by the specifics of the problem formulation.

While different studies have shown that demand forecasting depends on multiple factors and hence can be used in multivariate modeling, the univariate methods like ARMA and ARIMA [NW02, NSF17] might be sufficient for short term forecast. *Machine learning* (ML) and *Artificial Intelligence* (AI) methods based on neural networks [SJD16, Hu17], support vector machines SVM) [Chi13], and fuzzy logic [SBH05] were applied to capture complex non-linear relationships between inputs and outputs. When comparing ARIMA, traditional machine learning, and artificial neural networks (ANN) modeling, some recent articles provide contradictory results. In [HTL13], ARIMA achieves better results than ANN, while the study [Ger17] claims that ANNs perform slightly better than ARIMA methods. In our work, we construct a deep-learning based Convolutional LSTM forecasting model (a hybrid model with both Convolutional and LSTM layers). The Convolutional LSTM model works well on time series data as shown in [JLX18, JLS19] for activity recognition and in [XCW15] for forecasting rainfall. They are good in long term demand prediction, and indeed, automatically captures non-linear patterns.

In general, the quality and the prediction power of the models designed by using ML and AI methods critically depend on the quality and quantity of historical data. To create a

good forecasting model, several approaches have been developed in the literature. One such approach is an ensemble of multiple forecasting methods applied on the same time series data and a weighted average of their forecasts is used as a final result [WSH09]. In our work, we pursue a different approach by making use of the normalized data from multiple companies and train a single global model to make traffic predictions. This makes our method highly scalable.

3.7 Conclusion

In our work, we proposed *IoTelligent*, a tool that applies machine learning techniques to forecast the companies' traffic demands over time, visualize traffic trends, identify and cluster devices, detect device anomalies and failures. We showed that among the different neural network architectures, Convolutional LSTM model performed the best for demand forecasting. In order to avoid maintaining and upgrading tens (or hundreds) of models (a different model per company), we designed and implemented a novel, scalable approach, where a global demand forecasting model is built using the combined data of all the companies. This method was improved by normalizing the "contribution" of individual company data in the combined global dataset. We also introduced uncertainty intervals to the forecasts to provide better information to the users. To solve the scalability issues with managing the millions of devices, we designed and evaluated a novel technique based on: (i) autoencoders, which extract the relevant features automatically from the network traffic stream; (ii) DBSCAN clustering to identify the group of devices that exhibit similar behavior, in order to flag anomalous devices. The designed management tool paves the way the industry can monitor their IoT assets for presence, functionality, and behavior at scale without the need to develop device specific models.

CHAPTER 4

An Empirical Study of Deep Neural Network Explanation Methods

Explaining the inner workings of deep neural network models have received considerable attention in recent years. Researchers have attempted to provide human parseable explanations justifying why a model performed a specific classification. Although many of these toolkits are available for use, it is unclear which style of explanation is preferred by end-users, thereby demanding investigation. We performed a cross-analysis Amazon Mechanical Turk study comparing the popular state-of-the-art explanation methods to empirically determine which are better in explaining model decisions. The participants were asked to compare explanation methods across applications spanning image, text, audio, and sensory domains. Among the surveyed methods, explanation-by-example was preferred in all domains except text sentiment classification, where LIME's method of annotating input text was preferred. We highlight qualitative aspects of employing the studied explainability methods and conclude with implications for researchers and engineers that seek to incorporate explanations into user-facing deployments.

4.1 Introduction

In recent years, the explainability of deep neural network (DNN) models has come under scrutiny due to their black box nature [BCR97, DK17]. While it can approximate complex and arbitrary functions, studying its structure often provides little to no insight on the actual underlying mechanics. It is hard to look "into" the network and ascertain why specific features are selected over others during training. Although the impressive state-of-the-art performance

cannot be disputed, it is increasingly insufficient to place confidence in inferences without justification. In particular, when considering deployments where models generate high-stakes predictions, sensitive decisions often mandate a sufficient accompanying explanation. For example, "Robot Radiologists" now provide superior MRI and X-Ray image classification in comparison to the average trained human expert [Rea19]. A life-or-death diagnosis undeniably justifies the use of the best-performing model; however, it is unreasonable for either a patient or medical professional to simply accept an automated prediction at face value. With the inclusion of privacy regulations, including the GDPR "right to explanation," such an explanation is not only desirable, but also legally mandated [gdp18b].

While multiple definitions exist across the literature, we define an explanation as an image, text, or other visual aid that accompanies a prediction to offer intuition into the underlying reasons for the model output. Previous works have introduced several alternative techniques to provide such insight into a DNN model inference. Approaches span contrasting styles that focus on different model elements, e.g., the training dataset or the learned feature representations. Model-transparent approaches such as Grad-CAM++ [CSH18a] and saliency maps [SVZ13] highlight which particular input features triggered key activations within a model's weights. Model-agnostic methods such as LIME [RSG16], SHAP [LL17], and Anchor [RSG18] treat the model as a black-box and attempt to approximate the relationship between the input sample and the output prediction. Finally, example-based methods [KL17, KKK16, HCL19, CLT19] offer instances from the training dataset in an attempt to capture the relationship between a given test input and the underlying training data that contributed to the model's decision.

Given the diverse suite of available methods, a challenge arises in determining which explanation is best suited to a particular application domain. The notion of a satisfying explanation is dictated primarily by the target audience. In the case of a radiology diagnosis, this includes both the patient and medical professional. Despite each method offering distinct benefits to a model developer, the non-technical end-user can easily become overwhelmed; a succinct yet clear explanation is needed. Selecting the most appropriate method for a deployment is currently limited by a lack of comprehensive empirical information elucidating

the preferred explanation style according to the average end-user (i.e. individuals without machine learning expertise). Although there have been several works in the literature comparing and surveying different explanation methods [CPC19, GBY18, CTR17, HB20, WCM20, LT19], to the best of our knowledge we are the first to explore the preferences of the general population toward explanation methods across multiple input domains. This insight can additionally help steer the efforts of researchers in designing new methods and improving upon existing solutions.

Study Details. We performed a comprehensive Mechanical Turk study comparing six of the most popular explanation methods across four distinct application domains to determine which styles are most preferred in understanding DNN model decisions. The explanations used include LIME, Grad-CAM++, Anchor, SHAP, saliency maps, and explanation-by-example. Applications incorporated data inputs including image, audio, text, and sensory analysis. Each study question compared two explanation methods for a test input, and the participant was instructed to select the method that they considered to offer a better explanation, thus providing a relative ranking. Responses were filtered out if the questionnaire was completed too quickly or if a sufficient fraction (20%) of the validation questions were incorrect. Every method was fine-tuned with optimized hyperparameters such that each was portrayed in the best manner possible. As there were no explanation-by-example implementation readily available, we developed and offer an open-source library *ExMatchina* for use.

Study Results. 4970 validated responses were collected from 455 participants. For the image, audio, and sensor application domains, the *explanation-by-example* style was the preferred method in **89.6%**, **70.9%**, and **84.8%** of the responses, respectively, when it was an available option, surpassing other methods by a statistically significant margin when considering bootstrap confidence. In the text application domain, *LIME*'s highlighted text with annotated sentiment prediction was the preferred explanation method in **70.4%** of the responses when it was an available option.

Key Contributions. This paper's contributions can be summarized as follows. First, we provide a unification, comparison, and analysis of existing explainability approaches for DNNs across various applications and input domains. Second, we present and discuss the results of a

Mechanical Turk study[1] identifying the relative preference of explanation styles by an average non-technical end-user. Finally, we offer an open-source library *ExMatchina*[2], providing a readily available and widely applicable implementation of explanation-by-example.

4.2 Unifying Visual Explanation Methods Across Input Domains

Traditionally, DNN explanation frameworks emphasize their application over input domains that are naturally visualizable and understandable, e.g., image or text. However, DNNs are regularly applied across a wide class of domains, including time-series data (e.g., sensory data) that humans may struggle to reason about. Although most of the popular explanation methods are designed for images and text, prior works have shown that they can be successfully applied to time-series data [SAE19, GGG19, AS19].

Thus, we seek a unified representation such that an arbitrary input domain's explanation can be visualized and presented to end-users. In particular, we focus on the following representative dataset domains: image, text, audio, and sensory (ECG) data. These selected input domains are intended to provide insight into a subset of the popular DNN use cases. The unified representation supplies a common substructure to facilitate comparisons across multiple domains and enables a better understanding of the benefits of a particular approach.

4.2.1 A Unified Representation of Visual Explanation Frameworks

Figure 4.1 summarizes our approach towards unifying visual explanation methods across input domains for a given pre-trained model and a test input. We focus on visual explanations as they are the predominant means for conveying explanations, e.g., via text or image representations. Our unification does not encompass adaptive NNs and methods exploring the intralayer and interlayer statistical properties [VSB18, CVN20, ABT20].

A pre-trained DNN model F maps a set of input data $\mathbb{X}$ for a domain D to a set of labels

[1]https://github.com/nesl/Explainability-Study

[2]https://github.com/nesl/ExMatchina

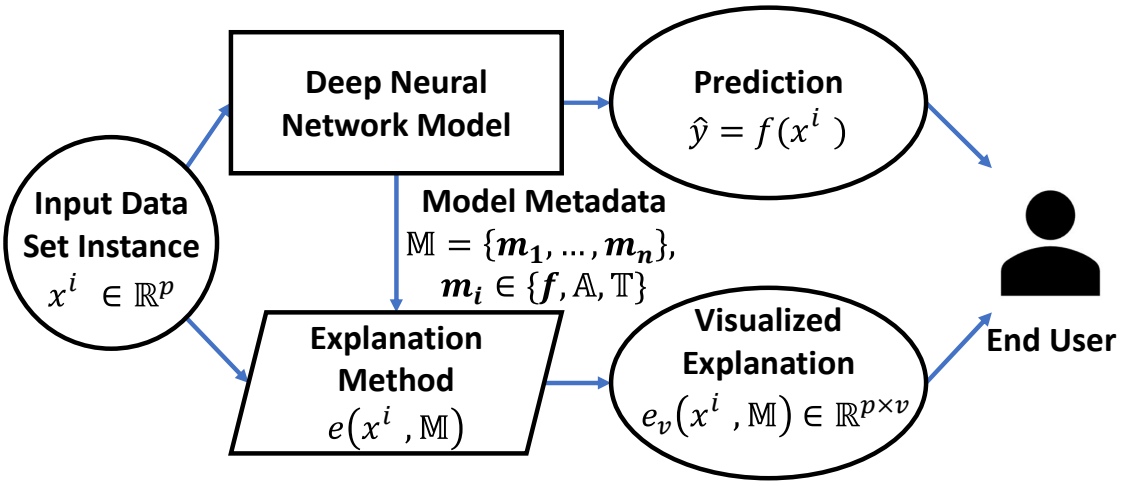


Figure 4.1: A unified representation of how we are drawing baseline comparisons for each visual explanation method, adapted from [CPC19].

$\mathbb{Y}$, i.e., $F : \mathbb{X} \to \mathbb{Y}$, where $\mathbb{X}_D = \{x_D^1, \ldots, x_D^n\}$ and the set $\mathbb{X}$ lies[3] in the domain space $\mathbb{R}^p$, i.e., $\mathbb{X} \in \mathbb{R}^p$. An explanation method requires a model metadata set $\mathbb{M} = \{m_1, \ldots, m_n\}$, where m_i represents a component of the DNN model that is utilized. Existing frameworks typically employ the set of activations $\mathbb{A}$, the training dataset $\mathbb{T}$, or the black-box representation of model f. Certain explanations may only require the black-box representation of the model or have access to the activations and training dataset as well, i.e. $\mathbb{M} \subseteq \{f, \mathbb{A}, \mathbb{T}\}$.

For a given input data instance x^i and a prediction $\hat{y} = F(x^i)$, an explanation method will generate an explanation $e(x^i, \mathbb{M})$. Traditionally, such an explanation would be in the form of a superimposed image on top of the original input data instance to highlight what features of the input "explain" the decision, i.e., both the explanation and the input were in the visual domain V and the space $\mathbb{R}^v$ such that $x_V^i \in \mathbb{R}^v$ and $e(x^i, \mathbb{M}) \in \mathbb{R}^v$. For any given task where the input domain is in a non-visual space, $\mathbb{X} \in \mathbb{R}^p \setminus \mathbb{R}^v$, the input and its explanation should be mapped to the visual space such that

$$e_v(x^i, \mathbb{M}) \in \mathbb{R}^{p \times v}. \tag{4.1}$$

[3]For the sake of clarity, we omit the notation D as we assume the input from all domains has been translated to the real space, e.g., a text input is first converted to an integer representation.

4.2.2 Superimposition Based Explanation Methods

A vast majority of explanation methods focus on visualizing an explanation by superimposing values onto the original test input data instance. While most of these frameworks already support the image and text domains, we also sought to provide visualized explanations for tasks in the audio and sensory domains. Generally, sensory and audio data are time-series that can be plotted for visualization. While DNN models and associated explanations are trained and inferenced on the raw time-series data, explanations are ultimately presented in visual form. That is, explanation outputs are inherently mapped to images. The value at each time step of a time-series data is analogous to pixels in the image and, therefore, the explanation methods can be applied by highlighting time instances or slices on the plots of the signal waveforms. Thus, we will describe how this intuition can lead to visualizations in multiple domains for both *model-agnostic* and *model-transparent* explanation methods. Generally, both categories will result in a superimposition of an explanation $e(x^i, \mathbb{M})$ onto an input instance x^i, i.e.,

$$e_v(x^i, \mathbb{M}) = g(x^i) + g(e(x^i, \mathbb{M})) \tag{4.2}$$

where g is a function that projects both the input space as well as the original explanation onto the visual space, e.g., a plot of an ECG input sample.

Model-agnostic methods. Model-agnostic explanation methods treat the model as a black-box, i.e., $\mathbb{M} = \{f\}$. The general approach is to approximate the relationship between the input and the output prediction. Surrogate methods such as LIME [RSG16] create a proxy model that is inherently interpretable to produce a local approximation of the relationship between a prediction and perturbed instances of the input data. Anchor [RSG18] uses a similar perturbation-based approach to approximate the relationship between the model's prediction and the input data via *high-precision if-then rules*. Likewise, SHAP [LL17] computes the contribution of each feature of an input test instance to the output prediction. In particular, SHAP uses a game-theory based approach by computing Shapley values for each feature.

Model-transparent methods. Model-transparent superimposition methods have full access

to the DNN pipeline and typically focus on the relationship between an input instance, an output instance, and the associated activations of the hidden layers, i.e., $\mathbb{M} = \{f, \mathbb{A}\}$. These methods similarly aim to highlight the features of the input that are important to the output classification.

Popular methods such as saliency maps [SVZ13] visualize the gradient of the output classification with respect to the pixels of the input image. Grad-CAM++ [CSH18a] superimposes a heatmap on the regions of important input features computed by the weighted gradients of an output classification with respect to the final convolutional layer of a CNN model.

For both model-agnostic and model-transparent methods, the raw data is provided to the DNN model and the associated explanation method without modifications[4]. Upon generating the explanation, the raw data will be visualized and the explanation is superimposed on the visualized representation.

4.2.3 Training Data Based Explanation Methods

In contrast to methods that project explanations onto the input space, techniques such as explanation-by-example fall under a category that project explanations across the underlying training data or other representative prototype examples. There is a wide variety of explanation by example frameworks that may focus on various aspects of the DNN model as well as the training dataset, i.e., $\mathbb{M} = \{f, \mathbb{A}, \mathbb{T}\}$. Protoype methods [PM18] and their associated critics [KKK16] focus on the distributions of the input dataset with respect to an inference to generate prototype examples for a particular input. Techniques such as influence functions [KL17] incorporate the relationship between the training dataset and the model weights at training time to identify the dataset samples that are most *influential* for a particular classification.

As explanation-by-example frameworks generate a set of examples as an explanation, the mapping of an explanation of any domain to the visual domain is more straightforward.

[4]Although an approach like LIME may require segmentation of the data, we assume these procedures to be subsumed by the function $e(x^i, \mathbb{M})$.

Explanation Style	Explanation Method	Domains			
		Image	**Text**	**Audio**	**Sensory data**
Superimposition over test input	LIME	✓	✓	✗	✗
	Anchor	✓	✓	✗	✗
	SHAP	✓	✓	❍	❍
	Saliency Maps	✓	✗	❍	❍
	Grad-CAM++	✓	✗	❍	❍
Explanation-by-Example	ExMatchina	✓	✓	✓	✓

Table 4.1: Comparing explanation methods across different input domains. Checkmarks (✓) indicate that a method is explicitly designed for a domain. Circles (❍) indicate that a method was able to be successfully adapted to a domain. Crossmarks (✗) indicate that adapting a method to the specified domain is non-trivial; we chose to exclude these evaluations to avoid potentially inaccurate representations that portray these methods in a suboptimal light.

Generated examples are visualized in the same manner by which an input data instance is visualized. Thus, if the general function for explanation by example frameworks is defined as

$$e(x^i, \mathbb{M}) = \mathbb{E} = examples(x^i), \tag{4.3}$$

then the associated visualized explanation for all domains will be

$$e_v(x^i, \mathbb{M}) = g(E), \forall E \in \mathbb{E}. \tag{4.4}$$

To the best of our knowledge, there is currently no openly available implementation of explanation-by-example. However, as this style of explanation has received considerable attention in recent work [PM18, KL17, KKK16, pro20], we believed it important to include in the study. Our open-source implementation of explanation-by-example, *ExMatchina*, provides the nearest matching data samples from the training dataset as representative examples. Nearest examples were selected by comparing feature activations at the last convolutional layer.

The nearest matching examples from the training set $\mathbb{T}$ is defined as the training data that has the highest cosine similarity with the test input x in their activations:

$$examples(x) = \max_{t \in \mathbb{T}} \cos(A^k(x), A^k(t)) = \max_{t \in \mathbb{T}} \frac{\sum_{i=1}^{n} A_i^k(x) A_i^k(t)}{\sqrt{\sum_{i=1}^{n} (A_i^k(x))^2} \sqrt{\sum_{i=1}^{n} (A_i^k(t))^2}} \quad (4.5)$$

where A^k represents the visualization of the k^{th} feature map as a vector of activation values. The use of cosine similarity as a distance metric for selecting the nearest neighbors is validated by Papernot and McDaniel [PM18]. For multiple examples, the $\max_N$ samples can be obtained.

We summarize the domains currently supported by state-of-the-art frameworks in Table 4.1.

4.3 Study Methodology

We conducted four separate Amazon Turk studies, one for each dataset. Study questions were formed in the following manner: first, a random test input and model-predicted class were presented along with two randomly selected explanations. Participants were asked to select which of the two available methods offered a better explanation for the provided model prediction. Due to the variability in the amount of time required to parse a test input and the associated explanations, the number of questions each participant answered varied across each study; the image classification questionnaire consisted of 15 comparisons, 12 for text classification, 6 for keyword classification, and 12 for the ECG-based arrhythmia heartbeat classification task.

Validating Responses. Two filtering criteria were included to eliminate participants providing illegitimate responses. First, participants that provided responses faster than a minimum threshold required to quickly read the survey were removed to exclude submissions auto-completed by bots. Second, each test input was accompanied by a validation question asking if they agree with the model prediction without providing the true label to cross-reference whether a participant was willing and able to comprehend the test input provided;

Task	Image Recognition	Sentiment Analysis	Key Word Detection	Heartbeat Classification
Domain	Image	Text	Audio	Sensory data (ECG)
Dataset	Cifar-10	Sentiment140	Speech Commands	MIT-BIH Arrhythmia
Classes	10	2	10	5

Table 4.2: An overview of the application tasks and datasets used in our study

those that failed 20% of these validation questions were eliminated from the published results. As the average participant is unlikely to have sufficient insight into the ECG sensor inputs, validation questions in that particular questionnaire offered the true label and simply asked participants to select whether or not the model was correct in its prediction.

IRB Exemption and Compensation. This research study has been certified as exempt from review by the IRB per 45 CFR 46.101, category 2 (UCLA IRB#20-000893). Participants were compensated at a rate of 15 USD per hour.

4.3.1 Tasks and Datasets

In an attempt to capture a wide array of common DNN use cases, we selected well-known classification datasets across each of the surveyed input domains, as depicted in Table 4.2. The test samples included in the study questionnaire were randomly selected from each domain's test set. The Cifar10 dataset [KH09] was selected for image data, the Sentiment140 dataset [GBH09] for text, Google's Speech Commands dataset [War18] for audio, and the the MIT-BIH Arrhythmia ECG Dataset [MM01] for sensory data. Cifar10 is a classic image classification dataset with 10 different classes. The text domain application performed sentiment analysis over Sentiment140, consisting of tweets with associated positive or negative sentiment; the shorter text length was explicitly desired to maximize participant engagement. For audio keyword recognition, the ten most common keywords of one-second long utterances were extracted from Google's speech commands dataset. In the sensory data domain, ECG data represents a class of sensory values that are relatively familiar and recognizable to the average individual. A heartbeat classification application was derived from the MIT-BIH

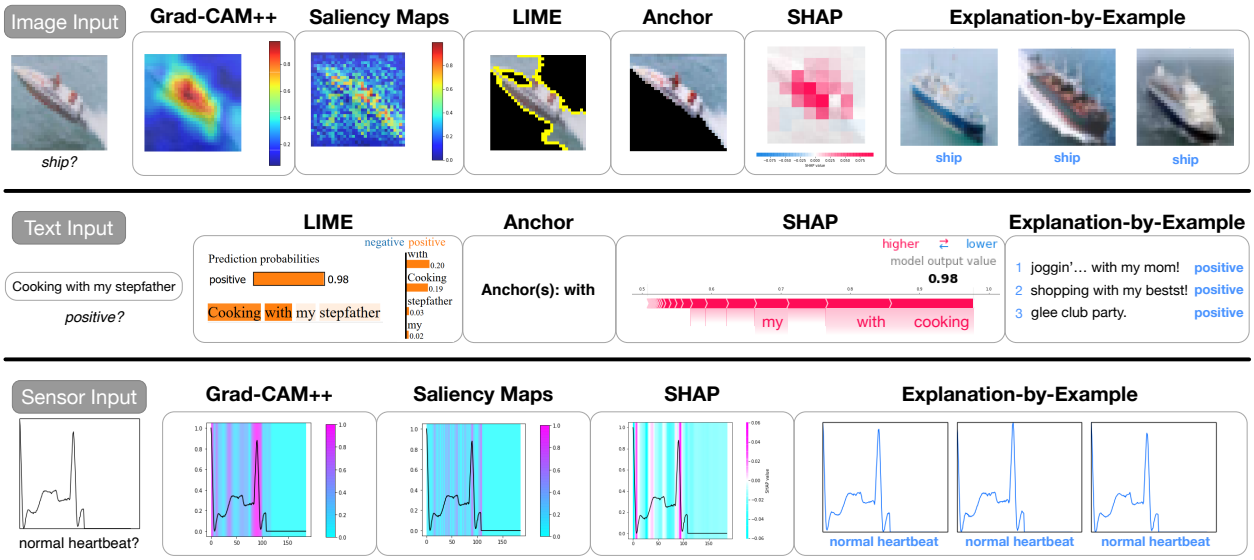


Figure 4.2: Depiction of surveyed explanation methods for image, text, and ECG input.

Arrhythmia Dataset. Tabular data were excluded from our study as models are typically derived using interpretable linear models or decision trees instead of DNNs.

4.3.2 Models and Explanations

A CNN (Convolutional Network Network) model was trained for each of the four tasks. Batch-normalization and dropout layers were used in all models. The trained models achieved f-1 scores of 87.8%, 83.5%, 90.2%, and 86.2% on Image classification, Text Sentiment Analysis, Key Word detection, and ECG Heartbeat classification, respectively. Screenshots of the explanations for image, text, and sensor data are presented in Figure 4.2. More specific details on the model architecture, hyper-parameters, and audio data explanations are available in the supplementary material.

4.3.2.1 Configuring and Optimizing Explanation Methods

To provide a comparative analysis across various approaches to visual explanations, we selected a subset of methods that were most commonly explored in related works and explainability studies [CPC19, CTR17, GBY18]: Grad-CAM++, saliency maps, LIME, Anchor, SHAP, and *ExMatchina*. Though several heatmap methods exist [FV17, PDS18, SCD17], we selected

Grad-CAM++ as the representative as it provides a more localized and compact explanation when compared to Grad-CAM, from which it is based on [CSH18a]. Our open-source implementation of explanation-by-example, *ExMatchina*, uses NN-query based on a number of prior works that advocate for a similar approach [GGG19, PM18].

Model transparent explanation methods including Grad-CAM++, saliency maps, and *ExMatchina* require the specification of which model layer to use in generating explanations. To ensure a fair comparison for these methods, we universally selected the activations after the last convolutional layer to uniformly capture the information extracted by the convolutional layers. This approach is validated by previous work [CSH18a] and our empirical observation that the last convolutional layers generally provided the best explanations.

LIME and Anchor [KVV19] required various hyperparameter tuning to generate acceptable explanations on images. The Felzenszwalb segmentation algorithm [FH04] was used to segment images into super-pixels.

SHAP has several different approximation methods to choose from. DeepExplainer was used on text, and GradientExplainer was used on image, audio, and ECG, as we empirically observed these to provide the best explanations for the datasets in our study.

Not all of the methods studied were intended for the suite of explored datasets. In particular, Grad-CAM++ and saliency maps are intended for image classification only. However, by treating the ECG and audio test samples as a one-dimensional image with only a single channel, normalized between 0 and 1, we were able to extract explanations for these datasets using these methods.

While we were able to successfully adapt the Grad-CAM++, saliency map, and SHAP methods for audio and sensory domains, it was non-trivial to adapt the saliency map and Grad-CAM++ to text and LIME and anchor to audio and ECG. To avoid potentially inaccurate representations that risk portraying methods in a suboptimal manner, we opted to exclude their application to these respective domains for this study.

Explanation toolkits that provided a built-in display method were used as is, and visual depictions were generated for those that did not provide such a solution. For example, as

SHAP was used on ECG and audio data, which cannot be illustrated in the exact same manner as an image, we created plots and highlighted the slices with the SHAP value gradients to illustrate the time slices that contributed positively or negatively. Similarly, in applying Grad-CAM++ and saliency maps to ECG and audio data, we highlighted the background of the time slices with the associated heatmap. In order to make audio data more comprehensible for the survey participants, we generated videos for each test sample and explanation. The waveform of the sample was plotted and overlaid with the audio track. When played, the video displayed a vertical bar that translates horizontally to accordingly indicate which part of the waveform is significant based on the particular explanation.

4.4 Study Results & Discussion

Table 4.3 presents the aggregated results of the Mechanical Turk study. After questionnaire validation, 4970 responses were collected across 455 individuals. Explanation-by-example was the largely preferred explanation style for image, audio, and sensory data classification; when *ExMatchina* was an available explanation, participants selected it 89.6%, 70.9%, and 84.8% of the time, respectively. In the text input domain, LIME was the preferred explanation method, as it was preferred 70.4% of the time. The presented confidence intervals are calculated using the bootstrap method as described in [DE96] with 95% confidence interval. These results should not be interpreted to conclude that any of these methods are inherently superior, but instead indicate that these are the methods most appealing to those who may not possess knowledge of machine learning, i.e. the "non-expert" layperson.

4.4.1 Usability and Stability of Explanations

LIME and Anchor were particularly unstable due to the inconsistent results generated across multiple runs. An additional contributor to their poor performance in the image domain was the reliance on segmentation algorithms to first split the image into superpixels. Selecting across the multiple available algorithms (e.g., Felzenszwalb, Slic, Quickshift, and Compact watershed) increased the overhead of ensuring the overall method was applied effectively.

Explanation Method	Image Study	Text Study	Audio Study	Sensor Study
LIME	47.7 ± 4.5%	**70.4 ± 3.6%**	-	-
Anchor	38.9 ± 4.3%	25.8 ± 3.5%	-	-
SHAP	33.7 ± 4.3%	59.9 ± 3.8%	34.7 ± 4.8%	32.8 ± 3.3%
Saliency Maps	39.4 ± 4.3%	-	46.1 ± 5.1%	40.4 ± 3.5%
Grad-CAM++	50.8 ± 4.5%	-	48.1 ± 5.3%	42.0 ± 3.5%
ExMatchina	**89.6 ± 2.6%**	43.7 ± 3.9%	**70.9 ± 4.7%**	**84.8 ± 2.5%**

Table 4.3: Results of the Mechanical Turk study evaluating user preference for DNN explanation methods across image, text, audio, and sensory input domains. Survey questions individually compare two methods at a time, with each explanation compared to all other available methods equally. Results indicate the rate by which users selected a particular method when it is an available explanation, with 95% bootstrap confidence intervals.

In contrast, segmentation was not necessary in the text domain as the granularity of words provide a natural unit of semantic value. Nevertheless, LIME and Anchor's explanations were still unstable across repeated runs; different explanations can be generated each time these methods are run on the same test sample. Moreover, each application required experimentation with different kernel and hyper-parameters settings to optimize against the generated explanations. In summary, these highly tunable methods incur increased configuration complexity and a higher probability of suboptimal specification.

In contrast, the other surveyed methods including Grad-CAM++, saliency maps, SHAP, and explanation-by-example did not require such fine-grained hyperparameters tuning, making them significantly easier to use. The only configuration parameter was the model layer to use in generating explanations. Moreover, these methods provide stable explanations; repeated runs over the same test instance lead to the same result.

4.4.2 Idealized vs Actualized Explanations

An individual's expectation of an explanation is sometimes different than those provided by the studied methods. In particular, superimposition explanations such as Grad-CAM++, saliency maps, and SHAP highlight input features that are most important to the predicted classification. However, these features may be in stark contrast to human intuition; similarly, Anchor and LIME would occasionally mask features that humans think are explicitly important. For example, given an image of an airplane in the sky, superimposition methods might highlight the sky to indicate that the sky is the most important detail determining that the object is an airplane; however, a human might prefer that the airplane is the defining characteristic leading to the airplane labeling. Similarly, when explaining audio data these methods would occasionally emphasize sections of the input that contained no speech (i.e., noise, background audio, or silence). In ECG data, flat sections may be highlighted instead of the actual heartbeat spikes. An ironic downside of using superimposition methods is that they show features which the models think are important, but which humans might not focus on, ultimately leading to the explanation method being generally less preferable.

Similarly, explanation-by-example should ideally produce examples that are always highly similar to the test input. In the image, audio, and ECG domains, we found that the nearest training examples repeatedly offered an intuitive and semantically similar mapping of training data, associated labels, and model inference. For example, nearest image examples commonly depicted the predicted class with a visually similar position, orientation, and relative sizing. Cases where the training examples could be considered dissimilar (i.e. "bad" explanations) were relatively rare. This consistency of explanation is a likely contributor to the preference of this method over the alternatives. In the audio and ECG domains, training examples were often impressively similar, including matching signal noise, event length, and position within the sample window. However, the semantic similarity of explanation-by-example is fundamentally limited by the quality of the training data; a lack of similar examples would necessarily lead to a subpar explanation [KKK16]. This was most obviously apparent in the text domain, where the nearest training examples to a test input may be seemingly unrelated.

In the text domain, LIME in particular offers an intuitive annotation of the text with associated sentiments and expected probabilities. The accuracy by which the annotated sentiments consistently matched human intuition, combined with its assembly into a natural visual interface, are likely key contributors to its success in explaining text data. Interestingly, LIME and Anchor occasionally assign sentiment to words that are neutral according to human intuition; for example, assigning positive or negative sentiment to words such as "at" or "with".

4.4.3 Privacy Risks

Unlike superimposition methods, explanation-by-example mandates access to a set of reference data (e.g., training data) by which to generate nearest examples. This poses privacy risks regarding potentially exposing personally identifiable information, particularly when operating over sensitive training data. Those seeking to employ explanation-by-example methods in deployments should ensure that the examples offered do not violate user privacy. In contrast, model-transparent methods that reveal model weights offer a reduced attack surface for inferring underlying data; model-agnostic methods that rely only on a test input and the model prediction function are able to mostly circumvent these concerns [SRS17]. In certain instances, differential privacy techniques can be used to anonymize training data when revealing training examples [ACG16, BWW19]. While we note that this may impact the quality of explanation, studying its quantitative effect is left to future work.

4.5 Conclusion

Successfully explaining deep neural network models hinges upon having an effective means of communicating the inner-workings of these complex processes. In certain domains, the explicit need for interpretable models outweighs the performance gains of black-box neural network methods. However, in the cases where the performance offered by DNNs are needed, it is important that they are accompanied by explanations that provide satisfying insight into model behavior.

Our study across hundreds of participants conclude that explanation-by-examples and LIME are the currently preferred explanation styles according to the average non-technical end-user. In input domains spanning visual, audio, and sensory data, explanation by nearest training examples offer users an opportunity to compare features across a test input and similarly mapped ground-truth examples. In the text domain, LIME's method of decomposing and annotating test inputs provides an intuitive visual approach to text classification. Although the other studied methods can be retargeted across many of the surveyed input domains, they failed to provide a more desirable explanation. Future efforts in designing DNN explanations are certain to challenge the current baseline; nevertheless, we hope our results and discussion bring insights empowering researchers and engineers to incorporate effective means of communicating complex inferences with the end-user.

CHAPTER 5

Automatic Concept Extraction for Concept Bottleneck-based Video Classification

Recent efforts in interpretable deep learning models have shown that concept-based explanation methods achieve competitive accuracy with standard end-to-end models and enable reasoning and intervention about extracted high-level visual concepts from images, e.g., identifying the wing color and beak length for bird-species classification. However, these concept bottleneck models rely on a necessary and sufficient set of predefined concepts–which is intractable for complex tasks such as video classification. For complex tasks, the labels and the relationship between visual elements span many frames, e.g., identifying a bird flying or catching prey–necessitating concepts with various levels of abstraction. To this end, we present CoDEx, an automatic Concept Discovery and Extraction module that rigorously composes a necessary and sufficient set of concept abstractions for concept-based video classification. *CoDEx* identifies a rich set of complex concept abstractions from natural language explanations of videos–obviating the need to predefine the amorphous set of concepts. To demonstrate our method's viability, we construct two new public datasets that combine existing complex video classification datasets with short, crowd-sourced natural language explanations for their labels. Our method elicits inherent complex concept abstractions in natural language to generalize concept-bottleneck methods to complex tasks.

5.1 Introduction

Deep neural networks (DNNs) provide unparalleled performance when applied to application domains, including video classification and activity recognition. However, the inherent

black-box nature of the DNNs inhibits the ability to explain the output decisions of a model. While opaque decision-making may be sufficient for certain tasks, several critical and sensitive applications force model developers to face a dilemma between selecting the best-performing solution or one that is inherently explainable. For example, in the healthcare domain ([YRJ19]), a life-or-death diagnosis compels the use of the best performing model, yet accepting an automated prediction without justification is wholly insufficient. Ideally, one could take advantage of the power of deep learning while still providing a sufficient understanding of why a model is making a particular decision, especially if the situation demands trust in a decision that can have severe impacts.

To address the need for model interpretability, researchers have sought to enable model intervention by leveraging concept bottleneck-based explanations. Unlike post hoc explanation methods–where techniques are used to extract an explanation for a given input for an inference by a trained black-box model ([CTR17, JNC20]), concept bottleneck models are inherently interpretable and take a human reasoning-inspired approach to explaining a model inference based on an underlying set of concepts that define the decisions within an application. Thus far, prior works have focused on concept-based explanation models for image ([KBB09, KNT20]) and text classification ([MKL20]). However, the concepts are assumed to be given a priori by a domain expert–a process that may not result in a necessary and sufficient set of concepts. For instance, for bird species identification, an expert may have collected wing color but failed to collect beak color as a concept though it might be important for classification. More critically, prior works have considered simple concepts with the same level of abstraction, e.g., visual elements present in a single image. For more complex tasks such as video activity classification, a label may span multiple frames. Thus, the composing set of concepts will have various levels of abstraction representing relationships of various visual elements spanning multiple frames, e.g., a bird flapping its wings. So it is impractical to identify all the key concepts in advance. Therefore, unlike the prior works, we aim to exploit the complex abstractions inherent in natural language explanations to extract the set of important complex concepts.

Research Questions. In summary, this paper seeks to answer the following research questions:

- How can a machine automatically elicit the inherent complex concepts from natural language to construct a necessary and sufficient set of concepts for video classification tasks?
- Given that a machine can extract such concepts, are they informative and meaningful enough to be detected in videos by DNNs for downstream prediction tasks?
- Are the machine extracted concepts perceived by humans as good explanations for the correct classifications?

Approach. This paper introduces an automatic concept extraction module for concept bottleneck-based video classification. The bottleneck architecture equips a standard video classification model with an intermediate concept prediction layer that identifies concepts spanning multiple video frames. To compose the concepts that will be predicted by the model, we propose a natural language processing (NLP) based automatic Concept Discovery and Extraction module, CoDEx, to extract a rich set of concepts from natural language explanations of a video classification. NLP tools are leveraged to elicit inherent complex concept abstractions in natural language. CoDEx identifies and groups short textual fragments relating to events, thereby capturing the complex concepts from videos. Thus, we amortize the effort required to define and label the necessary and sufficient set of concepts. Moreover, we employ an attention mechanism to highlight and quantify which concepts are most important for a given decision.

To demonstrate the efficacy of our approach, we construct two new datasets–MLB V2E (Video to Explanations) for baseball activity classification and MSR-V2E for video category classification–that combine complex video classification datasets with short, crowd-sourced natural language explanations for their corresponding labels. We first compare our model against the existing standard end-to-end deep-learning methods for video classification and show that our architecture provides additional benefits of an inherently interpretable model with a marginal impact on performance (less than 0.3% accuracy loss on classification tasks). A subsequent user study showed that the extracted concepts were perceived by humans as good explanations for the classification on both the MLB-V2E and MSR-V2E datasets.

Contributions. We summarize our contributions as follows.

- We propose *CoDEx*, a concept discovery and extraction module that leverages NLP techniques to automatically extract complex concept abstractions from crowd-sourced, natural language explanations for a given video and label–obviating the need to manually pre-define a necessary and sufficient set of concepts thereby reducing the annotation effort.
- We construct two new public datasets, MLB-V2E and MSR-V2E, that combine complex video classification datasets with short, crowd-sourced natural language explanations and labels.
- We evaluate our approach on the two complex datasets and show that the concept-bottleneck model attains high concept accuracies while maintaining competitive task performance when compared to standard end-to-end video classification models.
- We also augment the concept-based explanation architecture to include an attention mechanism that highlights the importance of each concept for a given decision. We show that users prefer the concepts extracted by our method over baseline methods to explain a predicted label.

5.2 Related Work

There is a wide array of works in explainable deep learning for various applications. This work focuses on the concepts-based explanations for video classification, and this section provides an overview of the existing literature for overlapping domains.

Concept-Based Explanations for Images and Text. A number of existing works consider concept-bottleneck architectures where models are trained to interact with high-level concepts. Generally, the approaches are multi-task architectures, where the model first identifies a human-understandable set of concepts and then reasons about the identified concepts. Until now, the applications have been limited to static image and text applications. [KNT20] used pre-labeled concepts provided by the dataset to train a model that predicts

the concepts, which is then used to predict the final classification. However, the caveat is that the concepts had to be manually provided. [GWZ19] and [YKA20] proposed approaches that automatically extract groups of pixels from the input image that represent meaningful concepts for the prediction. They were designed largely for image classification and extract concepts directly from the dataset. [KWG18] propose a post-hoc explanation method that returns the importance of user-defined concepts for a classification. In the mentioned works, the concepts have been limited to simple concepts and are not suited for complex tasks such as video classification where we have complex concepts that may span multiple frames with various levels of abstraction.

Explanations for Video Classification Other approaches have been considered to explain video classification and activity recognition. [CSH18b] applied GradCAM and GradCAM++ to video classification, where for each frame, the important region of the frame to the model is highlighted as a heatmap. [HPH20] extract both spatial and temporal explanations from input videos by highlighting the relevant pixels. However, these are post-hoc techniques that focus on explaining black-box models, whereas our approach enables concept-bottleneck methods for video classification that are intended to be inherently interpretable and intervenable.

Video Captioning. In recent years, there is a large number of works ([PYL17, GGZ17, WMZ18, YTW19, ZZC18, CJ21, YKC17]) on video captioning. While they also employ natural language techniques, these works are tangential to generating text explanations for classifications, since they are merely describing the video. Our model provides an explanation justifying the *classification* of the video. Similarly, the associated datasets such as MSR-VTT ([XMY16]) only have videos with ground truth captions that only describe the video without the context of a classification–which often results in concepts that do not pertain to a classification. For instance, in the MSR-VTT dataset, an example set of captions for a video labeled as an "animal" is "a black and white horse runs around" and "a horse galloping through an open field." These two captions have a superfluous set of a concepts that do not pertain to the "animal" classification, e.g., "there is a horse, and a horse is an animal." Although some captions may be useful, we chose to construct our own dataset to ensure the text explanations focused on explaining the classification. **Semantic Concept Video**

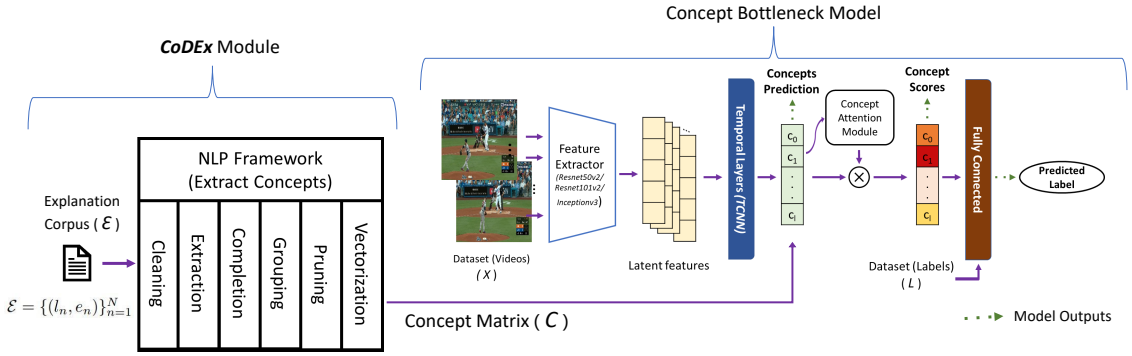


Figure 5.1: The overall pipeline showing the automatic concept extraction framework from natural language explanations and the concept bottleneck classification model training framework.

Classification. The closest works to this paper is the body of work in semantic concept video classification ([FLX04, FLG07]), where the concepts are defined as salient objects that are visually distinguishable video components. The concepts in these works are simple objects detected in the videos and are not complex enough to capture the semantics of events that happen over multiple frames of the videos. These works typically used traditional SVM-based video classifiers. [AZS14] represent a video category based on the co-occurrences of the semantic concepts and classify based on the co-occurrences, but their method requires a predefined set of concepts. Thus, we now present the methodology behind our automatic concept extraction for concept bottleneck video classification.

5.3 Concept Discovery and Bottleneck Video Classification

This work introduces *CoDEx*, an automatic concept extraction method from natural language explanations for concept-based video classification. Figure 6.3 depicts the overall concept-bottleneck pipeline, composed of *CoDEx* and the concept bottleneck model. *CoDEx* extracts a set of concepts from natural language explanations that will comprise the bottleneck layer for the video classification model. We first formalize the overall problem and then provide the methodology for both modules.

Problem Formalization. We assume that we have a training dataset $\{(x_n, l_n)\}_{n=1}^{N} = \mathcal{D}$

of videos x_n with a label $l_n \in \mathcal{L}$, where $\mathcal{L}$ is a predefined set of possible class labels for the video. Each video is represented as a sequence of frames $f \in \mathcal{F}$ where $\mathcal{F}$ is the set of video frames. Thus video $x_n = \langle f_{n0}, f_{n1}, \ldots, f_{nT} \rangle$, where f_{nt} represents frame t of video n. For each video x_n, we form a label-explanation pair (l_n, e_n), where e_n is a (short) natural language document explaining the given label l_n. If multiple annotators contribute to an explanation for video-label pair, (x_n, l_n) , then these are concatenated to form e_n. The full set of pairs $\mathcal{E} = \{(l_n, e_n)\}_{n=1}^{N}$ is the *explanation corpus*. Thus, the design goals are:

- **Concept Discovery and Extraction (CoDEx) Module:** Given the explanation corpus, first produce an $N \times K$ concept matrix, $\boldsymbol{C}$, where the (n, k)th element is 1 if the nth explanation contains discovered concept k and 0 otherwise. We call the nth row $\mathbf{c}_n$, the concept vector for video x_n. K is the total number of discovered concepts.
- **Concept Bottleneck Model:** Given a concept matrix, $\boldsymbol{C}$, the second goal is to train a concept bottleneck model such that for a given video x_i, we predict a concept vector $\mathbf{c}_i$–which indicates the presence or absence of concepts and their importance scores. The model then makes use of $\mathbf{c}_i$ to make the final video classification.

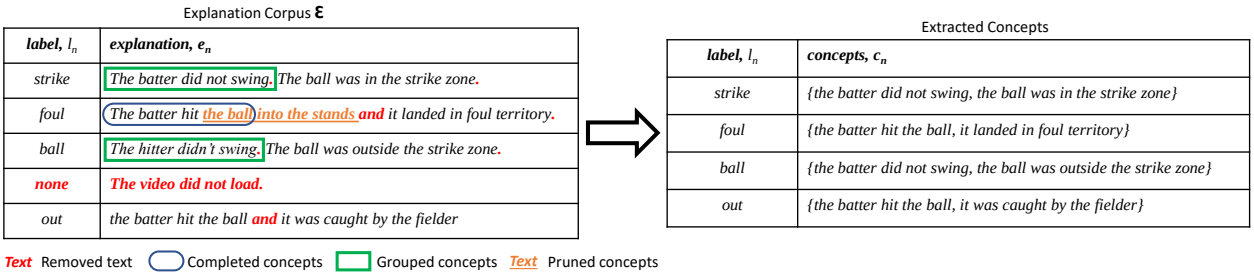


Figure 5.2: Running example for all six stages of the discovery pipeline module. The left table is the explanation corpus, with highlighted fragments to be modified. The right table contains the discovered concepts. The detailed step-by-step modifications are provided in Appendix A.1.

5.3.1 CoDEx: Concept Discovery and Extraction module

We now describe *CoDEx*, that extracts concepts from the explanation corpus, $\mathcal{E}$. The automatic extraction of the significant concepts is done in 6 steps, as outlined in Fig. 6.3. These are: **cleaning**, **extraction**, **grouping**, **completion**, **pruning**, and **vectorization**, which produce the final concept matrix, $\boldsymbol{C}$. Each of these steps are described below and illustrated with an example corpus depicted in Figure 5.2.

Cleaning. We remove explanations associated with corrupted or unlabeled videos from the explanation corpus. In Figure 5.2, this phase would remove the fourth row with the "none" label.

Extraction. The objective of this phase is to identify sentence constituents relevant to explaining the label. These text fragments, short sequences of words that appear in the document, are referred to as *raw concepts*. To achieve this, the cleaned explanation corpus is tokenized then passed through a pretrained constituency parser to recursively decompose the sentences. At each level of the constituency hierarchy, the text fragments are evaluated to determine whether they constitute a candidate raw concept. The rules for candidate raw concepts include the inclusion and exclusion rules mentioned in Table 5.1 and follow the widely adopted Universal POS tag naming convention for token types ([PDM12]). Every constituency parsed phrase that satisfies one of the two inclusion rules and not the exclusion rule is considered a candidate concept.

rule name	**rule**
Inclusion 1.	noun/pronoun → auxillary (optional) → particle (optional) → verb (optional)
Inclusion 2.	noun/pronoun → auxiliary whose lemma is 'be' → any token
Exclusion	subordinating conjunction

Table 5.1: Inclusion and exclusion rules for candidate concepts.

After the extraction process is completed, we have a set of raw concepts, $\widetilde{\mathcal{K}}$, and each video is associated with a subset of these raw concepts. An example of extracted raw concepts, $\widetilde{\mathcal{K}}$, can be found in Appendix A.1.

Completion. There are instances where the pretrained constituency parser will split sentences midway through a text fragment in one sentence that was kept whole in another. For instance, in Figure 5.2, the constituency parser splits the explanation for "foul" such that "the batter hit the ball" is incorrectly excluded from the raw concepts. To ensure that those concepts are captured, we perform a substring lookup of each raw concept through all documents of the explanation corpus and count an explanation as containing a raw concept if it contains the corresponding raw concept as a substring. This does not change the number of raw concepts identified but increases their frequency counts.

Grouping (similar raw concepts). When identical text fragments are identified in different explanations, they are counted directly as the same *raw concept*. However, we would ideally like to treat superficially different concepts as the same if they essentially carry the same meaning, e.g., Figure 5.2 highlights two different raw concepts that carry the same meaning and hence can be grouped. For this, we use agglomerative clustering ([Mul11]) approach that measures the degree of difference between pairs of raw concepts and groups them together if they are similar enough. Our key contribution here is the *distance metric* used in clustering which is a novel measure of meta-distance between raw concepts. This measures the difference between concepts based on two aspects of the raw concepts: their linguistic difference and their difference in terms of the label categories with which they are associated.

We define meta-metric, d, as combining a linguistic distance, d_{text} (capturing linguistic difference) as well as a meta-metric, d_{label} (capturing the difference in the labels associated with each raw concept). More formally, for two raw concepts $\kappa_i, \kappa_j \in \widetilde{\mathcal{K}}$ our distance is linear combination:

$$d(\kappa_i, \kappa_j) = d_{\text{text}}(\mathbf{v}_i, \mathbf{v}_j) + \lambda d_{\text{label}}(\mathbf{n}_i, \mathbf{n}_j) \tag{5.1}$$

where $\mathbf{v}_i$ is a sentence embedding for the text fragment of concept κ_i (e.g., based on the BERT model [DCL19]), d_{text} is a standard distance measure between vectors (e.g., cosine distance), d_{label} is a meta-distance which aims to capture the similarity between two label count vectors, and λ is a hyperparameter controlling the relative importance between textual and label distance. The inclusion of a label distance helps to distinguish between concepts that are

superficially linguistically very simlar, but have very distinct meanings within the domain of interest. For instance, without the d_{label}, the concepts "the ball passed inside the strike zone" and "the ball passed outside the strike zone" will be grouped together though they are very different concepts as they have a very small d_{text}. We provide a more formal definition of the meta-metric d_{label} more formally and provide some intuition behind its construction in Appendix A.2. We also exclude concept groups which occur rarely in the explanation corpus, with frequency less than some small threshold, t.

Pruning. Here, we seek a compact subset of concepts that, together, capture a high degree of information about the label while maintaining interpretability. More formally, after grouping, we have a set of raw concepts $\widetilde{\mathcal{K}} = \{\kappa_1, \ldots, \kappa_J\}$, and we seek some subset of maximally informative concepts $\mathcal{K}^\star = \{\kappa_{j_1} \ldots, \kappa_{j_K}\} \subseteq \widetilde{\mathcal{K}}$.

To see what is meant by *maximally informative*, consider a randomly selected entry in the explanation corpus (l, e). We define a binary random variable, C_j for each raw concept κ_j, and for any concept set $\mathcal{K} = \{\kappa_{j_1}, \ldots, \kappa_{j_K}\}$, random vector $\mathrm{C}_\mathcal{K} = [\mathrm{C}_1, \ldots, \mathrm{C}_K]$, such that $\mathrm{C}_j = 1$ if $\kappa_j \in e$ and 0 otherwise. Y is the random variable which takes label l.

We wish to choose the smallest subset of concepts such that the mutual information (MI)[1] between chosen concepts, $\mathcal{K}$, and label, Y, given by $I(Y; \mathrm{C}_\mathcal{K})$, is greater than a threshold fraction, $\gamma < 1$ of the MI between the label and the complete concept vector, $I(Y; \mathrm{C}_{\widetilde{\mathcal{K}}})$. That is to say we wish to find $\mathcal{K}$ which satisfies:

$$I(Y; \mathrm{C}_\mathcal{K}) \geq \gamma I(Y; \mathrm{C}_{\widetilde{\mathcal{K}}}) \tag{5.2}$$

and where there is no subset $\mathcal{K}' \subseteq \widetilde{\mathcal{K}}$ with $|\mathcal{K}'| < |\mathcal{K}|$ which also satisfies Equation equation 5.2. In practice, this is infeasible as the problem is combinatorial. However, we note that $f(\mathcal{K}) = I(Y; \mathrm{C}_\mathcal{K})$ is a monotone submodular set function of $\widetilde{\mathcal{K}}$. Given this, if we recursively construct a set of size K, by greedily adding single concepts that most improve the MI, the resulting set will be at least $1 - \frac{1}{e}$ as good as the most informative set of size K ([NWF78]). Therefore, we guarantee a highly-informative set $\mathcal{K}^\star$ by iteratively adding concepts to those

[1]We use the standard definition of mutual information (MI) for discrete random variables ([Mac03]).

previously selected, greedy with respect to the MI, until we have a set that satisfies Equation 5.2.

Vectorization. Each concept $\kappa_{j_k} \in \mathcal{K}^\star$ is given a unique index $k \in \{1, \ldots, K\}$, and each data-point, x_n is associated with a concept vector $\mathbf{c}_n = (c_{n1}, \ldots, c_{nK})$, where $c_{nk} = 1$ if $\kappa_{j_k} \in e_n$ and 0 otherwise, indicating the presence or absence of the kth concept in the nth explanation. The collection of all the concept vectors gives an $N \times K$ concept matrix, $\boldsymbol{C}$.

5.3.2 Concept Bottleneck Model

We use the videos, the extracted concepts from *CoDEx*, and the labels to train an interpretable concept-bottleneck model to predict the activity and the corresponding concepts. Figure 6.3 shows the overview of our bottleneck architecture. The activity label, the concepts, and the corresponding concept scores are the outputs of the interpretable model and are indicated by dotted arrows in Figure 6.3.

Our bottleneck model architecture is based on the standard end-to-end video classification models where we use convolutional neural network-based feature extractors pretrained on the Imagenet dataset [DDS09] to extract the spatial features from the videos. The features are then passed through temporal layers that can capture features across multiple frames which in turn is bottle-necked to predict the concepts. Lastly, we deploy an additive attention module ([BCB14] that gives the concept score α_c indicating the importance of every concept to the classification. The attention module also improves the interpretability of the the bottleneck model by indicating the key concepts for classification and this is evaluated in section 5.5. More details regarding the model architecture and hyper-parameters are in the Appendix A.4

Model loss function. The entire bottleneck classification model is trained in an end-to-end manner. Since the concepts are represented as binary vectors, we use sigmoid activation on the concept bottleneck layer and binary categorical loss function as the concept loss. The final layer of the classifier has softmax activations and categorical cross-entropy as the classification loss function. Thus, the overall loss function of the model is the sum of concept

loss and the classification loss. The hyperparameter β controls the tradeoff between concept loss, L_C, versus classification loss, L_Y as shown in equation 5.3.

$$Loss(L) = \frac{1}{N}\sum_{n=1}^{N}(L_{Y_n} + \beta \times L_{C_n}) \quad \text{where} \quad \beta > 0 \tag{5.3}$$

$$\text{and} \quad L_{Y_n} = -\sum_{j=1}^{m} y_j \log f_S(s_j)$$

$$\text{and} \quad L_{C_n} = \sum_{k=1}^{l}\left[-c_k^i \log(f_\sigma(s_k)) - (1 - c_k \log(1 - f_\sigma(s_k)))\right]$$

$$\text{and} \quad f_\sigma(s_i) = \frac{1}{1+e^{-s_i}} \quad , \quad f_S(s_i) = \frac{e^{s_i}}{\sum_{j=1}^{m} e^{s_j}}$$

Testing phase. Given an input test video, the model provides us with the activity prediction (label of the video), a concept vector indicating the relevant concepts that induced this classification and the concept importance score for each concept. By retrieving the phrase representing the concepts present in the video, the result obtained is a human-understandable explanation of the classification.

5.4 Implementation

To demonstrate our automatic concept extraction method, we construct two new datasets - MLB-V2E (Video to Explanations) and MSR-V2E, which combines short video clips with crowd-sourced classification labels and corresponding natural language explanations. For both datasets, we obtained a video activity label and natural language explanations for that label by crowd-sourcing on Amazon Mechanical Turk and used unrestricted text explanations to extract concepts automatically. For IRB exemption and compensation information, please refer to the Ethics Statement.

MLB-V2E Dataset: We used a subset of the MLB-Youtube video activity classification dataset introduced by [PR18]–which had segmented video clips containing the five primary activities in baseball: strike, ball, foul, out, in play. We preprocessed the dataset and extracted

2000 segmented video clips where each video was 12 seconds long, 224×224 in resolution, and recorded at 30 fps. To ensure that the quality of explanations is good, we screened over 450 participants. Based on their baseball knowledge, 150 participants were qualified to provide the natural language text explanations for our video clips. We have included a sample of our screening survey, the primary survey, and the explanations collected in the supplementary materials.

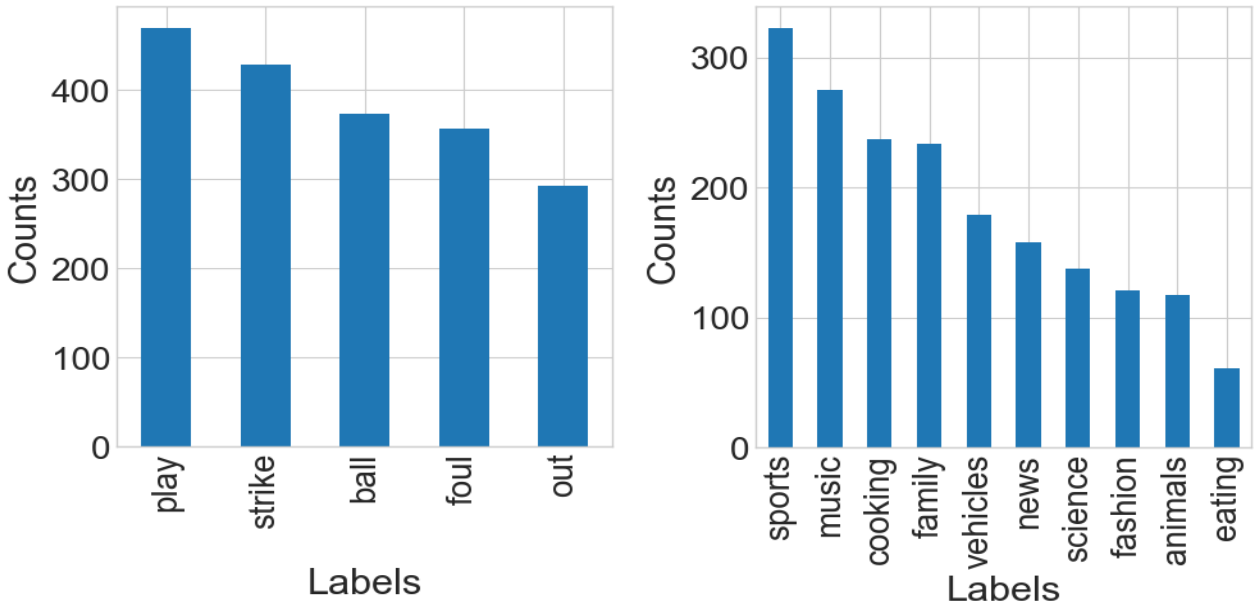
MSR-V2E Dataset: For this dataset, we used 2020 video clips from the MSR VTT dataset introduced by [XMY16]. The MSR-VTT dataset has general videos from everyday life and descriptions of these videos associated with them. Each video clip is between 10-30 seconds long, and approximately 200 participants provided the labels and explanations to construct the MSR-V2E dataset. The videos are classified into ten categories: Automobiles, Cooking, Beauty and Fashion, News, Science and Technology, Eating, Playing Sports, Music, Animals, and Family

Figure 5.3 shows the number of videos belonging to each category in the MLB-V2E and the MSR-V2E datasets. Since the MSR-V2E dataset is imbalanced, we do some weighted oversampling while training to ensure that the models learn to predict all the classes.

IRB Exemption and Compensation. This research study has been certified as exempt from review by the IRB and the participants were compensated at a rate of 15 USD per hour.

Dataset privacy. There was no personally identifiable information collected at anytime during the turk study. The responses provided by the mechanical turkers that are present in the dataset are completely anonymous.

Training: All our models were trained on 2 × Titan GTX GPUs using Adam optimizer. A summary of our entire model architecture and a trained model is provided in the supplementary materials.



(a) MLB-V2E Dataset (b) MSR-V2E Dataset

Figure 5.3: The number of videos belonging to each category on (a) the MLB-V2E dataset and (b) the MSR-V2E dataset

5.5 Results

Number of extracted concepts. Table 5.2 shows that the system extracted 78 concepts and 62 concepts from the explanation corpus of MLB-V2E and MSR-V2E respectively.

Dataset	Number of Concepts after each Phase			
	Extraction	Completion	Grouping	Pruning
MLB-V2E	1885	1885	225	78
MSR-V2E	1678	1678	104	62

Table 5.2: The number of concepts extracted by the Concept Discovery module from the explanation corpus after every phase.

The number of concepts remaining after the pruning phase is determined by the cumulative Mutual Information(MI) threshold. To identify the best threshold, we plotted the number of concepts at different thresholds versus performance of the model as shown in Figure 5.4. We found that the task classification performance did not increase after a certain number

of concepts and that optimal spot for the number of concepts corresponded to 90% Mutual Information.

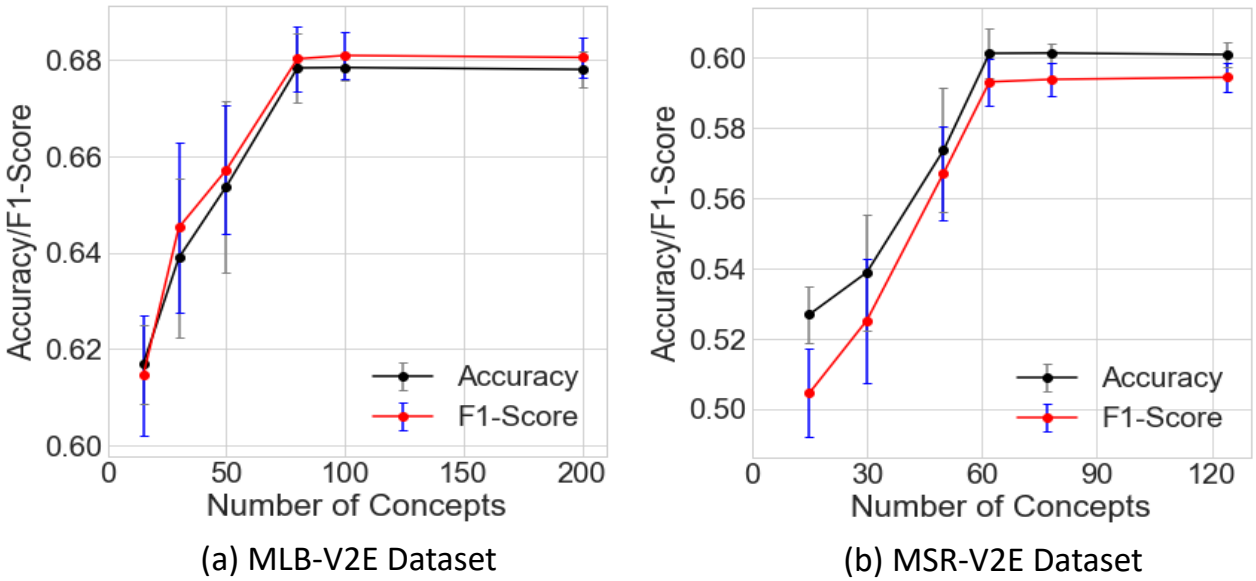


Figure 5.4: The number of concepts versus performance trade-off for the (a) the MLB-V2E dataset and (b) the MSR-V2E dataset.

Comparing concept-bottleneck models to baselines. We adopt model architectures and hyperparameters from standard well -performing approaches that fall under 3 categories: 1) without concept bottleneck [AA19], 2) with concept bottleneck [KNT20], 3) with concept bottleneck and attention. We compared the performance of models with the bottleneck layer with standard video classification models without a concept bottleneck layer. We find that, though the latent space was constrained to the limited set of concepts extracted from the explanation corpus, bottleneck models performed as well as the unconstrained models, on both datasets. We also find that the addition of the attention layer improves the concept prediction of the models. Table 5.3 shows that concept bottleneck models achieved comparable task accuracy to standard black-box models on both tasks, despite the bottleneck constraint while achieving high concept prediction performance. All the models in the table used Inception-V3 as their feature extractor. Appendix A.5 shows the performance with other feature extractors.

Ablation Study of CoDEx. We performed an ablation study to highlight the impact of each stage in CoDEx's pipeline. In the first experiment, we replaced the constituency

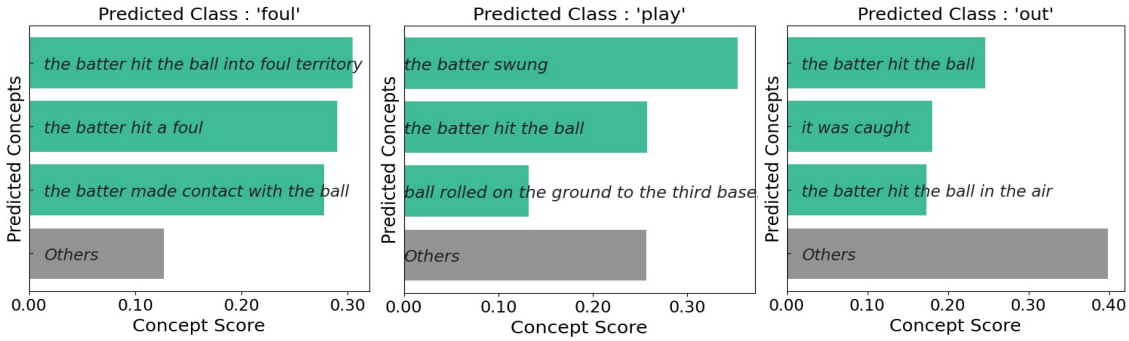
Dataset	Model Type	Task Classification		Concepts
		Accuracy(%)	F1-score	AUC
MLB-V2E	Standard	68.46 ± 1.27	0.68 ± 0.011	-
	CB	68.16 ± 1.12	0.68 ± 0.004	0.85 ± 0.003
	CB + Attn.	**68.38 ± 1.34**	**0.68 ± 0.004**	**0.88 ± 0.001**
MSR-V2E	Standard	61.79 ± 1.42	0.60 ± 0.012	-
	CB	61.42 ± 1.18	0.60 ± 0.013	0.83 ± 0.006
	CB + Attn.	**61.68 ± 1.23**	**0.60 ± 0.013**	**0.86 ± 0.004**

Table 5.3: Performance of Models with Inception V3 as the feature extractor. 'CB' refers to 'Concept Bottleneck'. The full table with results from different feature extractors can be found in Appendix A.5.

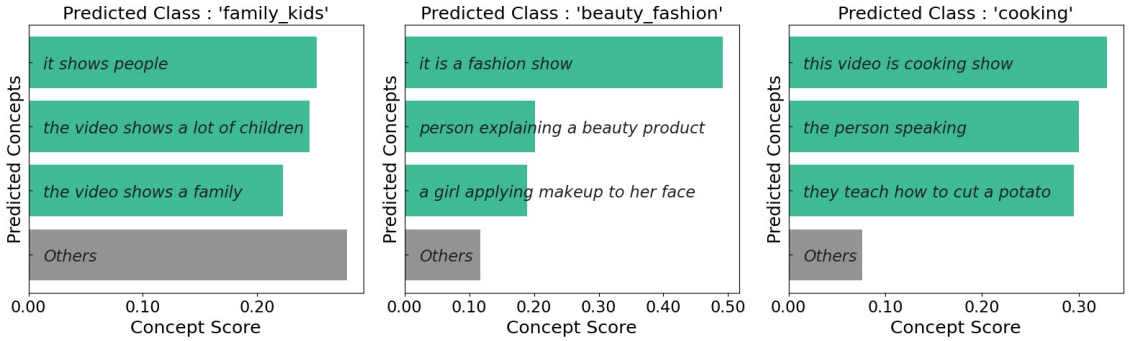
parser-based extraction phase with word-level tokens obtained from the explanation corpus as concepts. We found that using word-level concepts significantly reduced both the task and the concepts prediction performance. We also evaluated how removing the grouping and pruning phases from the pipeline would impact performance. We observed that the grouping and pruning stages had a greater impact on the concept prediction performance that affected the explainability of the model. Table 5.4 shows the results of our study.

Concept scores for interpretability. In addition to increasing the performance in concept prediction, the attention module also improves the explainability of the bottleneck model by providing an importance score for the concepts. Figure 5.5 shows the explanation from the concept bottleneck model with attention to test samples from the two datasets. The title shows the classification label, the y-axis indicates the top-3 concepts predicted as present in the video clip, and the x-axis corresponds to the concept score. *Others* refers to the sum of the importance of all the remaining concepts.

Human study to evaluate concepts' explainability. We performed a Mechanical Turk study to evaluate the explainability of our extracted complex concepts to the end-users. The participants were asked to select from four different options (presented in random)



(a) MLB-V2E Dataset



(b) MSR-V2E Dataset

Figure 5.5: Explanation offered by the model indicating the predicted class concepts present and their corresponding scores for (a) the MLB-V2E dataset (b) the MSR-V2E dataset.

Dataset	Concepts Extraction	Task Acc (%)	Task F1-score	Concepts AUC
MLB-V2E	CoDEx	**68.38**	**0.6802**	**0.8801**
	w/o Extraction	63.47	0.5823	0.8185
	w/o Grouping	67.80	0.6772	0.8122
	w/o Pruning	68.36	0.6802	0.8419
	w/o Grouping & Pruning	65.29	0.6526	0.7821
MSR-V2E	CoDEx	**61.68**	**0.6010**	**0.8600**
	w/o Extraction	58.02	0.5214	0.7830
	w/o Grouping	59.31	0.5745	0.7888
	w/o Pruning	61.68	0.6010	0.8131
	w/o Grouping & Pruning	54.70	0.5178	0.7467

Table 5.4: Ablation Studies: Shows the effect of each step in CoDEx on model performance

of what they consider to be the best possible Explanation for the classification of a given video. The four options are: Concepts predicted by bottleneck models without attention, Concepts predicted by bottleneck models with attention, Word-level concepts and a Random set of 2-5 concepts from the set of the most frequent concepts. The methodology of this study was inspired by [CGW09]'s paper. Figure 5.6 presents the aggregated results of the Mechanical Turk study. The complex concepts predicted by the concept bottleneck model with attention was considered as the preferred explanation by 68% and 57% of the responses in the MLB-V2E and MSR-V2E datasets respectively followed by the concepts bottleneck models without attention in 20% and 28% of the responses for the two datasets. The presented confidence intervals are calculated using the bootstrap method as described by [DE96] for 95% confidence.

Quantifying the Annotation effort. In prior works, annotating a dataset with concepts first requires identifying the necessary set of concepts that have to be annotated. Therefore, identifying the set of important concepts will either require a domain expert who has to

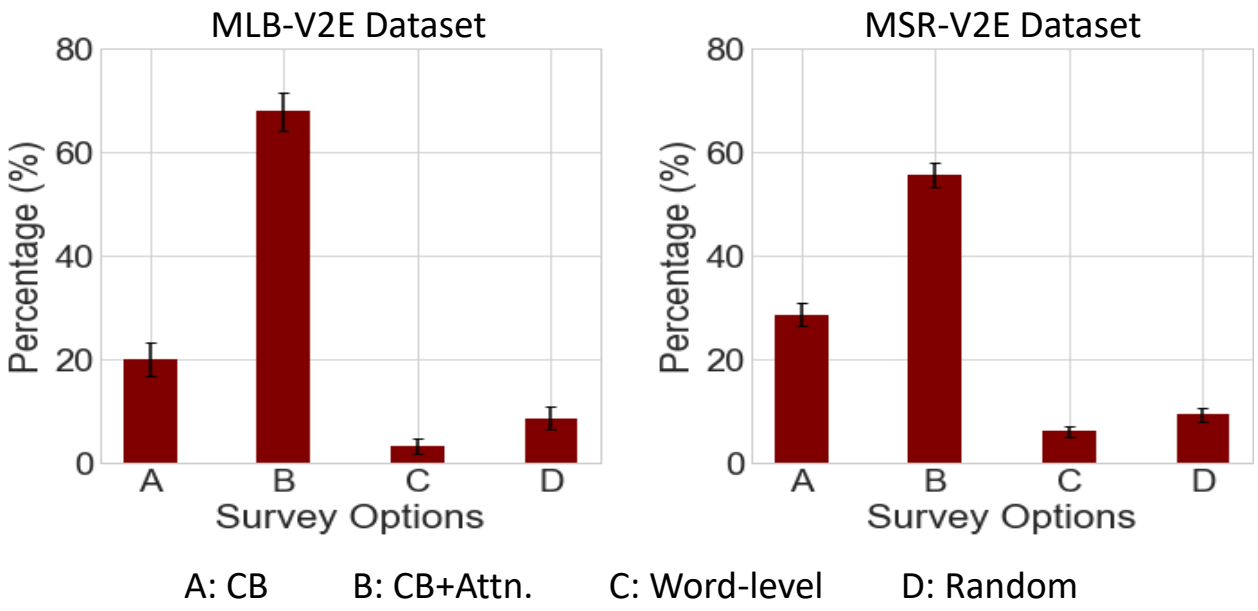


Figure 5.6: Survey responses with 95% bootstrap confidence interval for the two datasets. The participants preferred the concepts extracted by CoDEx and predicted by CB+Attn. as their preferred explanation by a significant margin.

provide a list of all the possible concepts or a separate study to identify the key concepts. Then annotating those concepts for each data point would require another study, making it more time-consuming and costly. Further, if the vocabulary of concepts is large, the annotation process would be inconvenient for the user. In our case, we significantly reduce the effort to identify and annotate the concepts with natural language explanations in a single user study. Moreover, natural language explanations can express richer compositions of concepts rather than simply identifying the presence or absence of an individual concept. To quantify the annotation effort, we conducted a small-scale study to annotate both MLB-V2E and MSR-V2E datasets with predefined concepts and compared it with the time it took to annotate with natural language explanations. For the MLB-V2E Dataset, we had a baseball domain expert who suggested thirty concepts to be annotated. For the MSR-V2E Dataset, we used the 62 concepts identified by CoDEx. The average time it took to complete per survey (each with ten videos) is indicated in Figure 5.7. We can see that the time it takes to annotate with text based explanations is 23.6% less for MLB-V2E Dataset and 30.8% less for MSR-V2E Dataset when compared to annotating with predefined concepts. We observe that

the time to annotate increases with the increase in the number of concepts to be annotated.

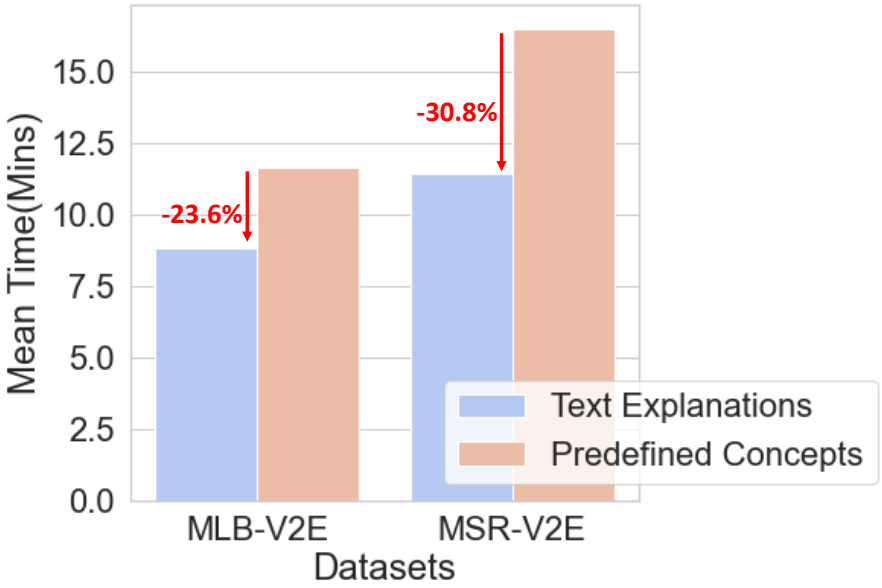


Figure 5.7: Average time taken to annotate 10 videos in a survey.

The helpfulness of extracted concepts in classification An MLP model trained on only the concepts (without video information) achieves a higher accuracy (7% for MLB-V2E and 4% for MSR-V2E) than the video classification model as shown in Table 5.5. This result indicates that the concepts extracted are meaningful for the classification task. Therefore, there is still headroom available if the concept bottleneck network can predict these initial concepts better from the videos.

Dataset	Task Accuracy(%)	Task F-1 score
MLB-V2E	75.68	0.7585
MSR-V2E	65.23	0.6422

Table 5.5: Performance of MLP classifier trained only on concepts (without videos)

The relationship between concepts and the classification task To understand the relationship between the extracted the concepts and the task classification, we compared the performance of the Concept Bottleneck models with a) MLP classifier b)Linear Classifier as the final classification layers. The results showed that there was approximately 2% drop in classification performance when using a Linear Classifier instead of an MLP as indicated in

Table 5.6. This indicates that, the classification task is not just a simple linear combination of the extracted concepts and the composition of concepts is important for classification.

Dataset	Accuracy (%)		Difference (%)
	MLP	Linear Classifier	
MLB-V2E	68.38	66.94	-1.44
MSR-V2E	61.68	60.02	-1.66

Table 5.6: Comparison of the performance of Bottleneck models with MLP classifier and Linear Classifier

5.6 Conclusion

The remarkable performance of deep neural networks is only limited by the stark limitation in clearly explaining their inner workings. While researchers have introduced feature highlighting post-hoc explanation techniques to provide insight into these black-box models, the inherently interpretable concept-bottleneck models offer a promising new approach to explanation by decomposing application tasks into a set of underlying concepts. We build upon concept-based explanations by introducing an automatic concept extraction module, *CoDEx*, to a general concept bottleneck architecture for identifying, training, and explaining video classification tasks. In coalescing concept definitions across crowd-sourced explanations, *CoDEx* amortizes the hardship of manually pre-defining the concepts while reducing the burden on the data annotators. We also show that our method provides reasonable explanations for classification without compromising performance compared to standard end-to-end video classification models.

CHAPTER 6

X-CHAR: Explainable Complex Human Activity Recognition via Temporal Concepts

Human activity recognition(HAR) utilizing multimodal sensors has emerged as a critical and high-impact research area in human-centered computing. Successful HAR applications have been demonstrated in the previous decade through academic research and industrial applications, including healthcare monitoring, smart home control, and daily sports tracking. However, compared to simple human activities, the expanding requirements of many current applications for complex human activity recognition (CHAR) have begun to attract the attention of the research field. Recent works have shown that deep learning models are highly accurate in predicting complex activities using sensor data. However, these deep learning models are black-box by design, and it is not easy to understand why they made a specific prediction. This results in a lack of trust in the model, hindering its deployment in the real world. In this chapter, we introduce *X-CHAR* , an eXplainable Complex Human Activity Recognition model that offers explanations in the form of human-understandable high-level temporal concepts. We evaluate our model on three different complex activity datasets and demonstrate that our model offers explanations without compromising the prediction accuracy. We also conducted a mechanical Turk study to show that the explanations offered by our model are more understandable than the explanations from existing methods for complex activity recognition.

6.1 Introduction

Any Artificial Intelligent (AI) systems often need to interact with humans, which are sentient beings (i.e., posse feelings) and often need an explanation or reassurance (regarding some topic or event). For instance, a bank, hospital, or any other organization that uses predictive analytics to decide on actions that impact people's lives should not make important decisions based on just a model's performance. Lack of trust in deep learning models has also hindered the adoption of AI systems in many fields [LSP06, Mil19]. Several governmental agencies are slowly proceeding to regulate AI to be more transparent. The first to move in this direction is the countries of the European Union, which have set several guidelines where they state that any AI-based system should be completely explainable [gdp18c, gdp18a]. With more transparent, explainable (i.e., interpretable) AI systems, the users will be better equipped to understand and, therefore, trust the services provided by the AI systems leading to broader adoption of such systems [MRC16, LGM20].

In some critical areas, the explainability of AI systems is a necessary component not only to understand the severity of the current situation but also to take further actions. For instance, in the healthcare domain, let us consider a Deep Neural Net (DNN) model performing diagnosis for a specific disease. Usually, a classic black-box DNN model merely outputs class labels. Even though it could outperform humans in sheer predictability, such inexplicable classification/prediction outcomes are useless in health domains. Sometimes, a doctor disagrees with the model's assessment. He/she needs to know why the model made specific outcomes to correct such mistakes in the future. On the contrary, it is also possible that the model most likely interprets some complex correlations from the data that the doctor is not aware of in the first place. Furthermore, who would be responsible given the misdiagnosis (e.g., a sick person is classified as healthy and does not get the proper treatment) caused by the outcomes of the black-box DNN model? This is still a gray area to be further decided by the policymakers.

Nowadays, DNN models are widely used for complex activity recognition tasks on different types of mobile sensor datasets (e.g., mobile phones, smartwatches, wearables) [JLX18, JLS19].

However, with the severity of use cases, it is crucial to explain the model's inference to the users. This can be climacteric in complex activity recognition tasks in critical domains (e.g., healthcare, autonomous cars, surveillance, and military domains). For example: using a neural network model to identify if a nurse is following a sanitary protocol before moving between patients—a task that spans long periods.

Usually, post-hoc explanations are not trustworthy [Rud19], which provide explanations that are not faithful to what the original model computes. On the other hand, saliency-based methods [SCD17, SVZ13] only indicate where the network is looking at and do not provide enough explanation to understand the interpretation of training data from any black-box models. As a consequence, a recent line of research has focused on developing interpretable deep learning models and providing explanations in the form of high-level human "concepts" [KNT20, JDC21]. For instance, the existing Concept Bottleneck Models (CBM) [KNT20] first predicts the presence of an intermediate set of human-specified concepts given the input instance. Then, the intermediate set of concepts is used to predict/classify the final output. However, for Complex activity classification, it is not enough for an explanation to merely indicate the presence and absence of concepts. This is because the ordering/sequence of a set of concepts and the frequency of these concepts matter to make the final complex activity inference. For example, for a nurse to perform a safe patient care activity, they must perform the following simple activities in sequence: sanitize hands-> check vitals -> blood collection -> clean patient -> sanitize hands. The key challenges in obtaining such explanations are as follows: *First,* the input sequence and the corresponding concepts can vary in length. *Second,* the ratio of the lengths of input sequence and concepts can also vary. *Third,* we do not usually have an accurate alignment (i.e., correspondence of the elements) of the input sequence and the concepts.

To this end, we propose X-CHAR (eXplainable Complex Activity Human Recognition), an end-to-end explainable deep learning model to classify the complex activity given an input sensor stream and provides an explanation that indicates the sequence and frequency of the concepts responsible for that classification. We consider that these complex activities are made up of simple activities occurring in a particular sequence, and these simple activities

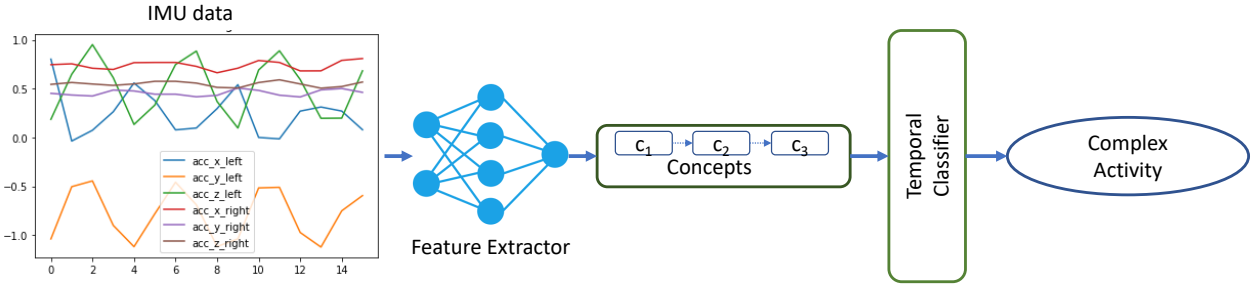


Figure 6.1: Our goal

are the temporal concepts. X-CHAR equips several standard deep-learning-based feature extractor layers with an intermediate *Temporal Concept Bottleneck* (TCB) layer to identify the temporal concepts. Since the input sequence and the concepts are unaligned and can be of variable length, then we propose a novel *Connectionist Temporal Classification* (CTC) loss to train the bottleneck layer and solve the challenges mentioned above. Thus, X-CHAR recognizes the complex activity and provides an explanation in the form of a sequence of high-level concepts for a given input data stream.

To demonstrate the efficacy of our approach, we use three complex activity datasets (Nurse Activity, Opportunity, and restaurant activities datasets) where the classification tasks require an inference explanation. We first compare our model against the existing standard end-to-end deep-learning methods for activity classification **add citations** and show that our architecture provides additional benefits of an inherently interpretable model without any impact on performance. We then conduct a user study to show that humans perceive the explanations (in the form of temporal concepts) as good explanations for the classification.

To summarize, the major contributions of this paper are as follows:

- We propose X-CHAR , an End-to-end explainable model architecture for complex activity recognition
- We show that having a bottleneck layer does not decrease the task accuracy compared to baseline models and has the additional benefit of being interpretable.
- We evaluate our approach on multiple time-series datasets. We conducted extensive experiments to show that adding this bottleneck layer provides explainability with no

loss in performance. We also conducted a human study to show that these explanations are preferred over other baseline explanation methods.

6.2 Definitions

Definition 1 (Simple Activity)**.** *A simple activity is defined as a unit-level activity that can be captured by a given sensor within a short window of time and cannot be broken down further given application semantics. The level of granularity depends on the sampling frequency of the sensor. For example, activities like 'pressing a switch' requires high sampling frequency while activities like sitting can be captured by low sampling frequency. Simple activities can either occur sequentially or concurrently.*

Definition 2 (Temporal Concepts)**.** *In explainable AI literature, the definition of concepts with regard to images is; a group of pixels that represent a higher-level and human-understandable feature that is relevant to a particular application/task. For instance, in bird species classification task, the wing and beak colors are considered as the concepts whereas the sky color or background is completely ignored for that application.*

In this paper, we define the temporal concept of the complex activity recognition as follows. A temporal concept is a section/portion of the input sequence that corresponds to a human understandable simple activity and is relevant to the particular task. Also, a group of temporal concepts have temporal dependencies, such that the sequence/order of the temporal concepts and the frequency of each temporal concept in the input sequence matter. Therefore, we can say that all temporal concepts are simple activities but not all simple activities are temporal concepts.

Definition 3 (Complex Activity)**.** *In this paper, a complex event is strictly defined as a particular pattern or sequence of $\geq$ 2 instances of concepts that have temporal dependencies. Under this definition, a complex event must be composed of multiple simple events that may evolve over long periods of time in different orders and frequency. For example, the "long-jump" sporting event is considered a complex event that consists of five sub-activities: "standing still", "running", "jumping", "landing", and "standing up". Another example is a*

sanitary protocol violation event in a hospital scenario: a nurse could violate the sanitary protocol if she or he processes one patient, and then processes another patient without proper sanitation.

6.3 System Design

In this section, we first formalize the overall problem and provide an overview of X-CHAR (Section 6.3.2) architecture. We then present the detailed descriptions of the different building blocks of the proposed system with appropriate figures, descriptions, and algorithms (as in Sections 6.3.3, 6.3.4, 6.3.5).

6.3.1 Problem Formulation

Support, there is a set of input time series instances X, where each instance $X_i = [x_{i,1}, x_{i,2}, x_{i,3}, .., x_{i,T}]$ and each time instance $x_{i,t} \in \mathbb{R}^{D\times 1}$. Here, T is the total number of time steps and D is the number of sensor channels. Given the multivariate time-series X, the tasks of the machine learning algorithm are two folds. *First,* each input instance X_i represents a sequence of concepts $\mathbf{c}_i$ such that $C \in \mathbb{R}^{K\times 1}$ (i.e., $K << T$) and the task is to find that sequence of concepts given the input instance X_i. Each concept represents a simple event that happens for a short period of time. *Second,* the sequence of concepts $\mathbf{c}_i$ represents a complex event $\mathbf{y}_i$ depending on the particular order of these concepts together. A complex event is composed of multiple simple events evolving over certain time periods in different orders and frequency. Therefore, the task is to utilize the concept sequence and frequency to classify the complex event $\mathbf{y}_i$ such that $\mathbf{y}_i \in Y$ and $Y \in \mathbb{R}^{N\times 1}$. Here N is the total number of complex activities in the dataset.

6.3.2 X-CHAR Model

This work introduces *X-CHAR* , an interpretable DNN architecture for the concept-based complex activity classification, that provides both the complex activity classification label and

it corresponding explanation in the form of temporal concepts for a given input sensory data. Figure 6.3 depicts the overall model architecture that shows how both the classification label and the corresponding explanation is generated, *X-CHAR* . There are three integral parts: Sensor fusion module, temporal bottleneck module, and classification module in X-CHAR . *First*, the Sensor fusion module extracts features from different sensors and maps them to a shared latent space. *Second*, the temporal bottleneck module extracts the temporal relationship between the obtained features and predicts the temporal concept associated with each timestep. *Third*, the classification module predicts the final complex activity label from the temporal concepts. Each of these modules are described below in detail.

6.3.3 Sensor Fusion Module

This module consists of two stages: the first stage extracts intra-sensor features and the second stage extracts the inter-sensor features.

6.3.3.1 Stage I

In *State I*, each sensor stream is considered separately and passed through a series of one-dimensional convolutional layers (1-D Conv). The 1-D convolutional layer is used for extracting local features from 1D patches in every sensor sequence to identify local patterns within the window of convolution. Since the same transformation is applied on every patch identified by the window. A pattern learned at one position can also be recognized at another position which makes 1D convolution layers translation invariant. The input to the first stage is input X_i which has the spape of $[D \times t]$. Here D is the number of sensory channels in the input stream. The output of the first stage is a collection of feature maps corresponding to each filter for every sequence, which is of the shape $[D \times t \times l_1]$. Here k_l is the number of 1-D kernels.

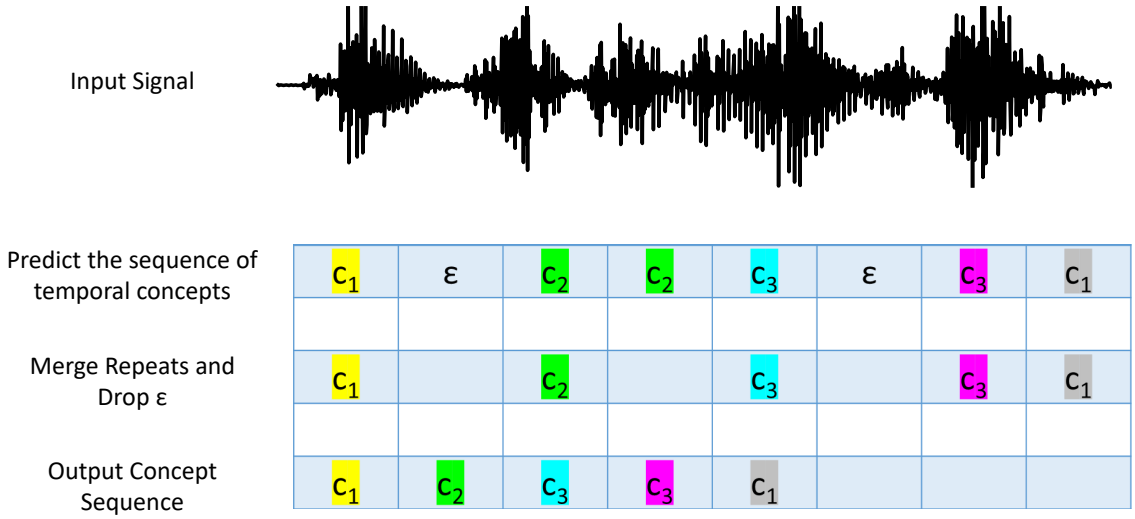


Figure 6.2: Temporal Concepts Decoder

6.3.3.2 Stage II

In the *Stage II* of the sensor fusion module, the proposed model captures the correlation across the corresponding axes in different sensors. Therefore, *X-CHAR* concatenates the feature maps to the shape of $[t \times (s \times l_1)]$ and feeds it to the second set of convolutional layers. The output of the second convolutional layer is of the shape $[t \times l_2]$. Here l_2 is the number of convolution kernels in *Stage II*.

6.3.4 Temporal Bottleneck Module

Recall that we mention three main challenges in Section I. The total numbers of input sequence and corresponding concepts and the ratio of their total numbers is not fixed. The exact accurate alignment (i.e., correspondence of the elements) of input sequence and the concepts is also unknown. To solve these challenges, we design the concept bottleneck model so that it can facilitate the proper alignment between input sequence and output concepts.

This module is designed to identify the sequence of concepts C_i from the given input X_i. There are three main components in this module: a set of bi-directional LSTM layers, a dense layer, and a softmax activation layer. First, this module uses a set of bi-directional LSTM layers to capture the temporal information from the preceding sensor fusion module. Next,

the time-distributed dense layer is used to map the temporal features from the proceeding module to different concepts to aid the softmax activation layer. The total number of neurons k in the dense layer is equal to the set of available concepts k in the dataset. Finally, the softmax layer outputs the probability distribution over the possible concepts at each timestep t.

Connectionist Temporal Classification Use Connectionist Temporal Classification (CTC) loss to model the concepts. The CTC algorithm can assign a probability for any Y given an X. It is alignment free, monotonicity with many to one mapping.

$$p(C_i|X_i) = \sum_{a \in A_{X_i,C_i}} \prod_{t=1}^{T} p_t(a_t|X_i) \tag{6.1}$$

where (X_i, C_i) is the ith pair of input and concept sequence, A_{X_i,C_i} is set of the alignment between (X_i, C_i) and $p_t(a_t|X_i)$ is the probability of alignment of a_t given the input instance X_i.

$$\mathcal{L}_C = -\log p(C|X) \tag{6.2}$$

6.3.5 Classifier

This module predicts the final complex activity. In this module, we use a Temporal Convolutional Neural network (TCN) layer followed by a dense layer with softmax activation. The TCN captures the relationship between the temporal concepts from the bottleneck module and the dense layer predicts the complex activity. Dense layer has 'n' neurons corresponding to the number of output classes. And since this is a classification problem we use softmax (σ) as the activation of the final dense layer which is given in Equation 3.2. Softmax function is used to impart probabilities to the logits 'a' when we have multiple classes and we get the probability distribution of output classes. We consider the most probable occurrence with respect to other outputs as the predicted class.

Classification Loss (L2): We use categorical cross entropy as our complex activity loss

function since it is a multi-class classification problem.

$$\mathcal{L}_{Y_n} = -\sum_{j=1}^{m} y_j \log f_S(s_j) \tag{6.3}$$

$$\text{where,} \quad f_S(s_i) = \frac{e^{s_i}}{\sum_{j=1}^{m} e^{s_j}} \tag{6.4}$$

Overall loss. The entire X-CHAR classification model is trained in an end-to-end manner. Therefore, the overall loss of the model (L) is a weighted sum of concepts loss (Lc) and classification loss (Ly) as written in Equations equation 6.3 as

$$\mathcal{L} = \beta \mathcal{L}_{Y_n} + (1 - \beta)\mathcal{L}_{C_n} \tag{6.5}$$

6.3.6 Inference Phase

After we've trained the model, we to use it to predict the complex activity and find the likely concept sequence as explanation for a given input. The predicted activity is obtained by taking the argmax of the probabilities from the classification layer and the corresponding concepts sequence is obtained by decoding the output of the temporal bottleneck module with beam search algorithm [LL04] and is show in Figure 6.2

6.4 Experimental Evaluations

6.4.1 Datasets

In this section, we first present the implementation details of a real-world prototype of the proposed system. Nextly, we evaluate the performance of the prototype in two of many realistic scenarios. Then, we introduce the comparison methods and metrics used in the evaluations. Finally, we discuss the performance evaluations in terms of figures and tables with proper descriptions.

Complex Nursing Activity Dataset. The complex nursing event dataset is based on a public dataset from Nursing Activity Recognition Challenge [LAT19]. The dataset contains

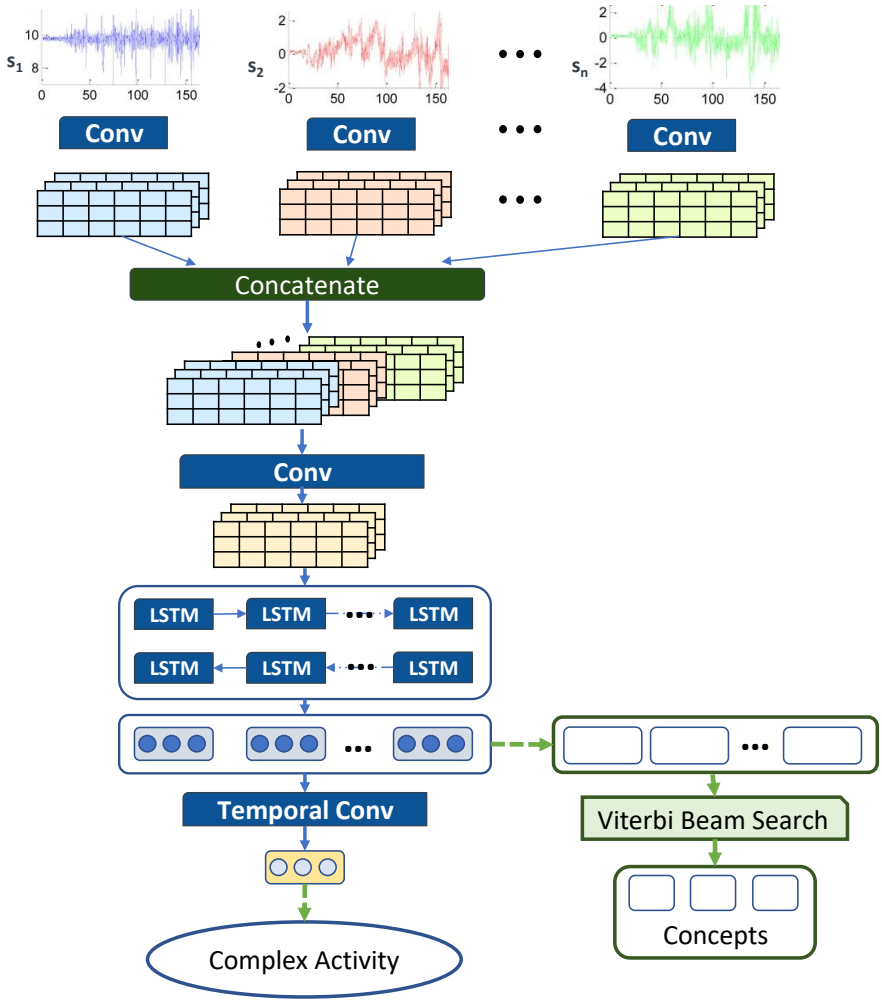


Figure 6.3: The overall model architecture

nurse activity data collected from three sources: Motion Capture, Meditag, and Accelerometer sensors, and it includes six different simple activities performed by eight subjects (nurses). The simple activities are (i) Vital signs measurements, (ii) Blood collection, (iii) Blood glucose measurement, (iv) Indwelling drip retention and connection, (v) Oral care, and (vi) Diaper exchange and cleaning of the area. Each of these activities is performed five times by each nurse. The duration of each data segment ranges from 30 to 60 seconds. These simple activities are considered as the concepts in our experiments. Because of the noisy and missing data problem in Motion capture and Meditag data, we only used the accelerometer data in our experiment. The sampling frequency of the accelerometer is 4Hz. The dataset is split into two parts: the first part contained data from 6 nurses, and the second part contained

Complex Nurse Activity	Concept Sequence
Physiological Measurement	Vitals → Blood collection → Blood Glucose
	Blood collection → Vitals → Blood glucose
Patient cleaning	Vitals → Oral care → Clean
	Oral care → Vitals → Clean
	Oral care → Blood glucose → Clean
	Blood glucose → Oral care → Clean
Unsanitary Operations	Clean → Blood glucose → Vitals
	Clean → Oral care → Vitals
	Vitals → Clean → Oral care
	Oral care → Clean → Blood glucose
Safe IV/Drips Procedure	Vitals → Blood collection → Drips → Vitals
	Vitals → Drips → Vitals
	Vitals → Drips → Blood collection → Vitals
Unsafe IV/Drips Procedure	Vitals → Blood collection → Drips
	Blood collection → Drips → Vitals
	Drips → Blood collection → Vitals

Table 6.1: Description of Complex Nurse Activity Dataset

data from the remaining two nurses. We then generate five complex nursing activities data by randomly selecting segments corresponding to the different concepts and concatenating them together in a predefined order for each complex activity as mentioned in Table 6.1. We generated a training dataset of 3000 complex activity samples (600 for each complex activity) from the first part and a validation dataset of 1000 complex activity samples (200 for each complex activity) from the second part. The results reported in Section 6.5 are based on the models' performance on the validation dataset.

Opportunity Dataset

This dataset had four trials from five different participants. Each trial had data from so many different sensors including body sensors, object sensors, and ambient sensors. Since

we focus on activity recognition task we consider the five inertial body-worn sensors which were worn in five different locations on the body: Left lower arm (LLA), Left upper arm (LUA), Right lower arm (RLA), Right Upper arm (RUA) and Back of the torso. These inertial units recorded accelerometer, gyroscope and magnetometer values. The activities performed had two different hierarchy of labels– High-level activities which considered the 4 major complex activities(Early morning routine, making sandwich, making tea, cleaning) and Low-level activities which had 17 different micro activities such as opening and closing doors, shelves, drawers, drinking etc.. In our evaluation, we preprocessed the dataset to remove nan values and segments without complex activity labels. We considered data from four users in training and the fifth user in testing.

CRAA: Complex Restaurant Activities from Audio dataset

This complex activity dataset this generated by using a subset of the audio samples from ESC-50 [Pic] and Kitchen-20 [MOF19] audio datasets. The ESC-50 dataset is a labeled collection of environmental audio recordings that consists of 5-second-long recordings organized into 50 semantical classes. Kitchen20 contains 5 to 10 second audio recordings from kitchen activities for 20 different classes. From these datasets, we considered the audio clips from 12 different categories as our concepts. We then constructed a audio-based complex human activity recognition dataset by concatenating the concepts is different sequences to obtain the various complex events. In particular, we generated the following 5 complex activities that would be performed by a person working in a restaurant: Making a juice, Making a Puree/sauce, Having a drink, Hygienic use of restroom, Unhygienic use of restroom. In total, we synthesized a training dataset of 1000 complex activity audio samples (200 for each complex activity) and a test dataset of 250 complex activity audio samples (50 for each complex activity). The brief description of the dataset is shown in Table 6.2

6.4.2 Baseline Complex Activity Detection Models

We compare the performance of X-CHAR to three existing state of the art complex activity prediction DNN models. They are:

Restaurant Worker Activity	Concept Sequence
Using Restroom (Hygienic)	Footsteps → Using toilet → Toilet flush → Wash Hands
	Wash Hands → Using toilet → Toilet flush → Wash Hands
Using Restroom (Unhygienic)	Footsteps → Using toilet → Toilet flush
	Footsteps → Wash Hands → Using toilet → Toilet flush
	Wash Hands → Using toilet → Toilet flush → Footsteps
Making a fruit juice	Open Shelf → Chopping → using blender
	Peeling → Chopping → using blender
	Open Shelf → Peeling → using blender
Making a puree/sauce	Peeling → Using blender → Chopping → using blender
	Chopping → Using blender → Peeling → using blender
	Open Shelf → Using blender → Chopping → using blender
Having a drink	Take glass → Pour water → Drink

Table 6.2: Description of CRAA: Complex Restaurant Activities from Audio dataset

6.4.2.1 DEBONAIR [CLP21]

It uses convolutional layers to extract features and LSTM layers to predict the complex activities. This model only predicts the final activities and not the concepts.

6.4.2.2 AROMA [PCY18]

The input data is split into a predefined smaller segments of equal size. Then each segment is predicted a simple activity and then a classifier predicts the complex activities from simple activities. This model requires the data to be labelled for every timestep.

6.4.2.3 Concept-Bottleneck Models [KNT20]

Concept bottleneck models(CBM) predict concepts that are provided at training time, and then using these concepts to predict the label. They only indicate the presence and absence of concepts and do not capture the temporal relations between them.

6.4.3 Baseline Explanation Methods

We compare X-CHAR explanations with other explanation methods. There are plenty of saliency-based post-hoc explanation methods that provide an explanation in the form of a heatmap highlighting the important portion of the input sample. In our paper, we chose GradCAM since it is one of the most widely used explanation technique and unlike other methods like LIME, SHAP, it doesn't involve splitting the input data into smaller windows which affects the quality of explanation.

6.4.3.1 GradCAM [SCD17]

Gradient-weighted Class Activation Mapping (Grad-CAM), uses the gradients of the target class flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the input for predicting that class.

6.4.3.2 Explanation by Examples [JNC20]

Explanation by Examples select particular instances of the dataset to explain the behavior of machine learning models. The nearest matching examples from the training set as chosen from the training dataset that has the highest cosine similarity with the given test input in their activations.

6.4.3.3 Concept-Bottleneck model [KNT20]

These are interpretable models that provide explanations in the form of highlevel human "concepts" that accompanies a prediction.

Dataset	Model	Task F1-score	Concept	
			Accuracy	Edit Distance
Nurse Activities	DEBONAIR	0.9691	-	-
	AROMA	0.9677	72.25	2.26
	CBM	0.9244	88.40	-
	X-CHAR	**0.9698**	**89.35**	**0.15**
Opportunity	DEBONAIR	**0.8382**	-	-
	AROMA	0.8318	53.97	2.87
	CBM	0.7726	64.52	-
	X-CHAR	0.8357	**67.36**	**1.54**
CRAA	DEBONAIR	0.9880	-	-
	AROMA	0.9576	86.40	1.94
	CBM	0.9032	96.58	-
	X-CHAR	**0.9886**	**97.70**	**0.10**

Table 6.3: Performance comparison of X-HAR with other baseline models on the three datasets

6.5 Results

6.5.1 Comparison of Model Performance

We compared the performance of *X-CHAR* with other complex activity recognition models discussed in Section 6.4.2 on the three datasets. We find that *X-CHAR* achieves the best F1-score on two of the datasets and best concept prediction accuracy on all three datasets. DEBONAIR is good in complex activity classification but it cannot predict concepts. AROMA has a decent task classification performance but a poor concept prediction accuracy. CBM models have good concept prediction accuracy but a poor classification performance since they don't capture the sequence information of the concepts.

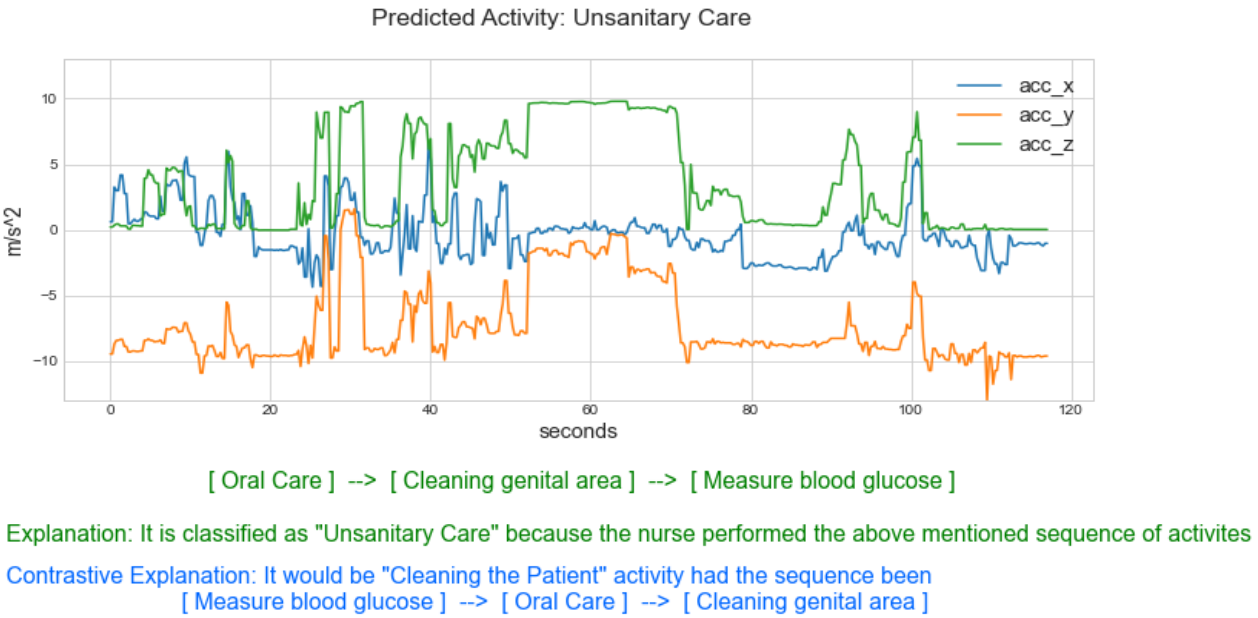


Figure 6.4: Explanation by *X-CHAR*

6.5.2 X-CHAR Explanations

In addition to the high performance in complex activity prediction, the temporal bottleneck module improves the explainability of *X-CHAR* by providing the sequence of occurrence of the temporal concepts. Figure 6.5 shows the explanation from *X-CHAR* to test samples from the nurse activity dataset. The title shows the classification label, and the explanation is provided below the plot in green.

6.5.3 Human Study to evaluate Concepts Explainability

We performed a Mechanical Turk study to evaluate the explainability of the temporal concepts from *X-CHAR* to the end-users. The participants were asked to select from four different options (presented in random) of what they consider to be the best possible Explanation for the classification of a given input sensor data. The four options are: Heatmaps from Gradcam, Explanation-by examples, Concepts predicted by bottleneck models and Temporal Concepts predicted *X-CHAR* . The methodology of this study was inspired by [CGW09]'s paper. Figure 6.6 presents the aggregated results of the Mechanical Turk study. The temporal concepts predicted by the *X-CHAR* was considered as the preferred explanation by 82% of

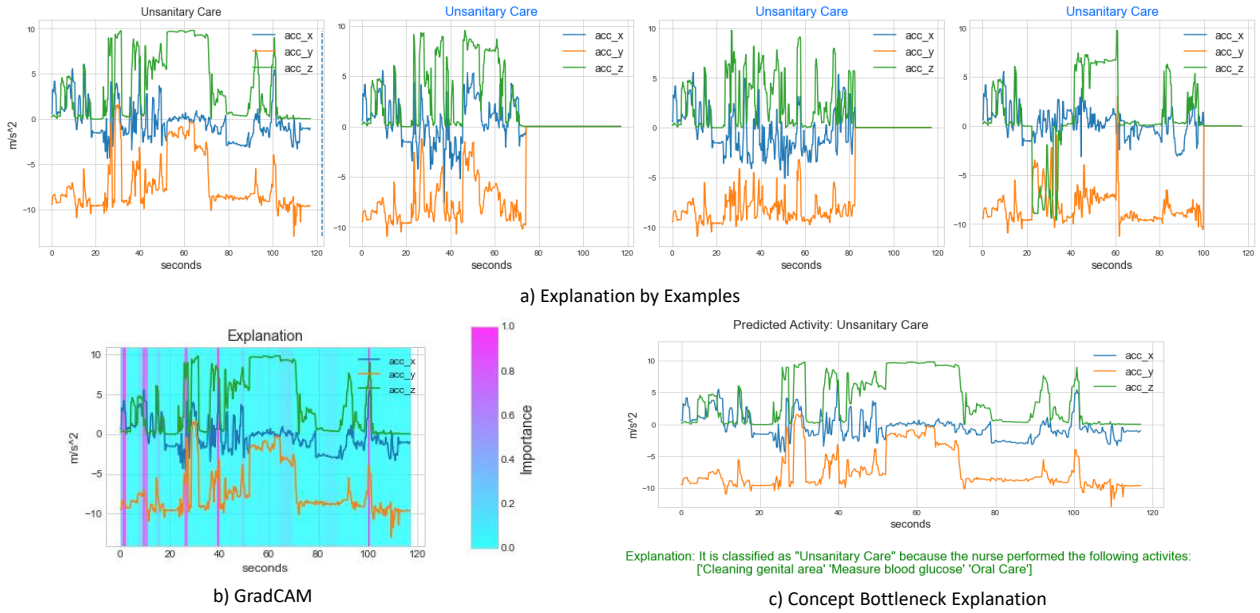


Figure 6.5: Explanation from other methods that were included in the survey

the responses in the Nurse complex activity dataset followed by the concepts bottleneck models in 54% of the responses. The presented confidence intervals are calculated using the bootstrap method as described by [DE96] for 95% confidence.

6.6 Discussion

Sensory data, unlike images or text, is inherently difficult to understand for humans. Our results have confirmed the same and has shown that humans prefer explanations in the form of concepts especially when the input data is inherently complex. As the number of channels in the input increased, more study participants preferred temporal concept based explanations. A possible future direction to explore is to replace the classifier module after the temporal bottleneck module with a Neurosymbolic layer. This might help the model to learn rules from the temporal concepts thereby resulting in better explanations.

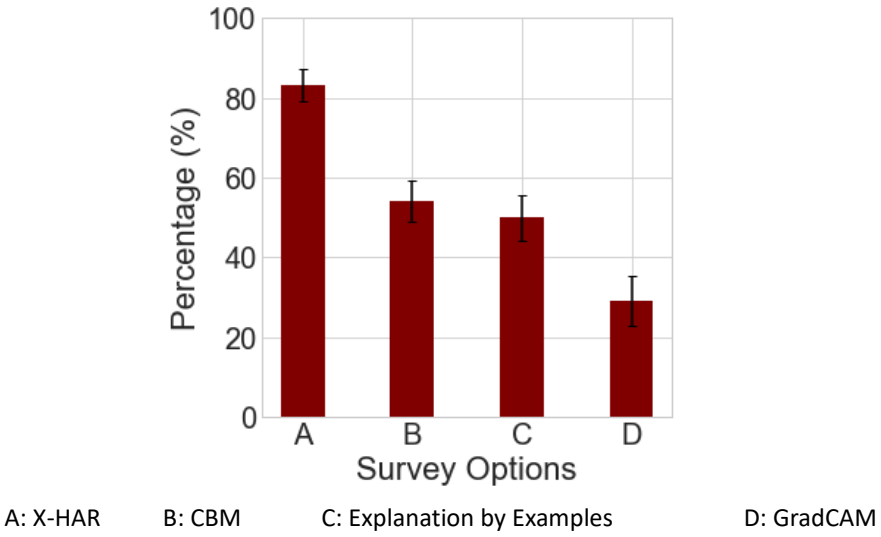


Figure 6.6: Survey Results

6.7 Conclusion

In this paper, we presented *X-CHAR* , an interpretable deep learning model for complex activity prediction that provides the sequence of simple activities relevant to the task called temporal concepts as explanation. We thoroughly evaluated our model on three complex activity datasets and showed that our model achieves state-of-the-art performance in complex activity recognition while also providing human-friendly explanations.

CHAPTER 7

Discussion and Future Work

7.1 Explanation by examples offers robustness to Adversarial Attacks

In addition to explaining the model's prediction on benign data, we also apply explanation by example method to adversarial examples. Deep learning models are vulnerable to adversarial attacks [LQX20]. The goal of the experiment is to study how does explanation-by-example behave on adversarial test inputs. We test the robustness of this method by first testing on adversarial inputs generated by existing methods.

Existing adversarial attacks Consider the input image $\mathbf{x} \in \mathbb{R}^N$ where $N = 3 \times W \times H$ is the size of images. The hard-label classifier gives its predicted label y, where $y \in \{1, ..., C\}$. Currently, given the original image $\mathbf{x}^{org}$, its ground-truth label y^{org}, a target class $t \neq y^{org}$ and adversarial image $\mathbf{x}^{adv}$, one of the commonly used methods for generating an adversarial image is to minimize the L_2-distance:

$$\underset{\mathbf{x}^{adv} \in \mathbb{R}^N}{\text{minimize}} \qquad \|\mathbf{x}^{adv} - \mathbf{x}^{org}\|_2^2 \tag{7.1a}$$

$$\text{subject to} \qquad f(\mathbf{x}^{adv}) = t \tag{7.1b}$$

By minimizing the L_2-distance, the attacker seeks to make the adversarial images inconspicuous from the human perspective such that the attack cannot be easily prevented [GMP17] [MGF17]. The existing adversarial attacks relies on solving the above optimization problem to create adversarial examples.

7.1.0.1 Experiments

We conducted experiments using the CIFAR-10 dataset where we apply the CnW attack [CW17], SignOPT attack [CSC19] and our adaptive versions of these attacks and produce explanations by explanation by example method . The experiments show that explanation by example identifies adversarial examples generated by the exiting algorithms where the attacker doesn't have any knowledge of the explanation method being used. As Figure shows, though the adversarial example is classified to a different target class, the nearest neighbours in the latent space still belong to the correct class thereby indicating that the input image has been tampered with.

7.2 Types of input modalities

As we have shown in this dissertation, the input modality decides the preferred explanation type for an end-user. We can broadly classify the inputs into the following two categories:

7.2.0.1 Inherently Interpretable Inputs

Data from input modalities like audio and video are considered to be inherently interpretable because an average human can label or identify the image or audio at given time. Hence, the concepts for the data of this category can be obtained by using a turk study when the mechanical turkers will provide a written explanation and concepts can be extracted using NLP techniques.

7.2.0.2 Inherently Non-Interpretable Inputs

Sensory time series data whose relationships with the prediction labels are less intuitive and more subtle, and thus not easily understood. This inherent non-interpretability of the inputs makes it hard for those who are not experts in sensor technologies to have a causal understanding between physical, physiological, and cognitive states and raw sensor measurements. This type of input data requires a person with domain expertise to label the

Figure 7.1: CnW and SignOPT are existing adversarial attacks and all attack examples are mis-classified as horse though appear to be a frog or car. Explanation by example still shows frog and car images as the nearest examples even though they are misclassified adversarial inputs.

concepts and hence are challenging to obtain.

7.3 A Hybrid Explanation Method for Multimodal Systems

In this dissertation we identified the preferred type of post-hoc explanation method as well as designed inherently interpretable neural networks. A possible direction for future research would be to combine both inherently interpretable models and post-hoc methods and provide an hybrid explanation to the end-user such that they receive the benefits of both the worlds. For example, we can predict the temporal concepts using *X-CHAR* model and use saliency based methods to highlight the important concepts for prediction.

CHAPTER 8

Conclusion

This dissertation proposes the framework that is robust to the heterogenetiy of the devices and provides trusthworthy predictions accompanied with user-friendly explanations. First, we identified the different challenges and heterogeneities in the sensing modalities and discussed our solution, SenseHAR– a virtual activity sensor that is robust to the variations and availability of the inertial sensors using deep learning. SenseHAR enables the inferring of human activities even when the set of available devices varies over time by mapping the information from the available devices to a shared low-dimensional latent space. We explain the method to calibrate the SenseHAR for a new device in the absence of labeled data with the help of an existing calibrated device. Finally, we do a comprehensive evaluation of SenseHAR and compare its performance with the state-of-the-art multimodal architectures for various device combinations and configurations. We also provide a scalable device management framework to identify anomalous IoT devices.

Then, to provide human understandable explanations for classification tasks we conducted a study involving tasks across multiple domains. Our study with hundreds of participants concluded that explanation-by-examples and LIME are the currently preferred explanation styles according to the average non-technical end-user. In input domains spanning visual, audio, and sensory data, explanation by nearest training examples offer users an opportunity to compare features across a test input and similarly mapped ground-truth examples. In the text domain, LIME's method of decomposing and annotating test inputs provides an intuitive visual approach to text classification. Next, we focus on inherently interpretable models particularly on models that are interpretable via concepts. We first proposed CoDEx, an NLP-based automatic concept extraction module that can extract concepts from text and

decreases the annotation effort. Finally, we designed an interepretable DNN model, *X-CHAR* , that can capture the temporal relation of the concepts unlike the prior works. We showed that our model achieved state-of-the-art performance in complex activity recognition and provided human-friendly explanations for multi-modal inputs.

APPENDIX A

CoDEx Module

A.1 Running Example to describe the Concept Discovery and Extraction Pipeline

Example (A simple Explanation corpus)**.** *Consider using baseball domain with a few meaningful labels, a null label* $\mathcal{L} = \{strike, ball, foul, out, none\}$ *and five entries in the explanation corpus* $\mathcal{E}$*:*

id, n	***label***, l_n	***explanation***, e_n
1	*strike*	*The batter did not swing. The ball was in the strike zone.*
2	*foul*	*the batter hit the ball into the stands and it landed in foul territory*
3	*ball*	*The hitter didn't swing. The ball was outside the strike zone.*
4	*none*	*The video did not load.*
5	*out*	*the batter hit the ball and it was caught by the fielder*

Step 1 (After Cleaning Phase)**.** *The none label entry is removed after the Cleaning phase*

id, n	***label***, l_n	***explanation***, e_n
1	*strike*	*The batter did not swing. The ball was in the strike zone.*
2	*foul*	*the batter hit the ball into the stands and it landed in foul territory.*
3	*ball*	*The hitter didn't swing. The ball was outside the strike zone.*
4	*out*	*the batter hit the ball and it was caught by the fielder*

Step 2 (After Extraction Phase)**.** *The raw concepts are extracted based on the defined rules discussed in Table 5.1 corresponding to each explanation with the help of a pre-trained constituency parser.*

id, n	***label***, l_n	***raw concepts***, $\widetilde{\mathcal{K}}$
1	*strike*	*The batter did not swing, The ball was in the strike zone*
2	*foul*	*the ball into the stands, it landed in foul territory*
3	*ball*	*The hitter didn't swing, The ball was outside the strike zone*
4	*out*	*the batter hit the ball, it was caught by the fielder*

Step 3 (After Completion Phase)**.** *The concept 'the batter hit the ball' was not extracted by the Extraction phase for the Id* 2 *explanation in this corpus. This missing concept is retrieved through the sub-string matching.*

id, n	***label***, l_n	***raw concepts***, $\widetilde{\mathcal{K}}$
1	*strike*	*The batter did not swing, The ball was in the strike zone*
2	*foul*	*the batter hit the ball, the ball into the stands, it landed in foul territory*
3	*ball*	*The hitter didn't swing, The ball was outside the strike zone*
4	*out*	*the batter hit the ball, it was caught by the fielder*

Step 4 (After Grouping Phase)**.** *We show the grouped concepts for this example corpus*

Concept-id, i	***Concept groups***
1	*[The batter did not swing, The hitter didn't swing]*
2	*[the batter hit the ball, the batter hit the ball]*
3	*[The ball was in the strike zone]*
4	*[The ball into the stands]*
5	*[it landed in foul territory]*
6	*[The ball was outside the strike zone]*
7	*[it was caught by the fielder]*

Step 5 (After Pruning Phase)**.** *The concept 'the ball into the stands' of Concept-Index 4 from previous table was not contributing much and hence was pruned.*

Concept-id, i	***Concept groups***
1	*[The batter did not swing, The hitter didn't swing]*
2	*[the batter hit the ball, the batter hit the ball]*
3	*[The ball was in the strike zone]*
4	*[it landed in foul territory]*
5	*[The ball was outside the strike zone]*
6	*[it was caught by the fielder]*

Step 6 (After Vectorization)**.** *After pruning, each sample is mapped to their corresponding concept vector. The value of concept vector at index i is* 1 *if e_n has the concept with index i, else, it's* 0*. The Matrix containing all the concept vectors is called the Concept Matrix, $\boldsymbol{C}$*

id, n	***label***, l_n	***Concept Vector***, $\mathbf{c}_n$
1	*strike*	*[1,0,1,0,0,0]*
2	*foul*	*[0,1,0,1,0,0]*
3	*ball*	*[1,0,0,0,1,0]*
4	*out*	*[0,1,0,0,0,1]*

A.2 Meta-distance for label based proximity

At the end of the **Completion Phase** we define a count for each raw concept, $\kappa_i \in \widetilde{\mathcal{K}}$, given by M_i. And for each label category $l \in \mathcal{L}$, we define a label count for raw concept κ_i as m_{il} where i is the index of the concept. These count the presence of raw concepts explanations, and $\sum_{l\in\mathcal{L}} m_{il} = M_i$. Finally we group together raw concept κ_i's label counts into a label count vector $\mathbf{m}_i = [m_{i1}, \ldots, m_{i|\mathcal{L}|}]$

Now we describe the meta-metric d_{label} used in the **Grouping** phase more formally and provide some intuition behind its construction. Consider that we have two raw concepts $\kappa_i, \kappa_j \in \widetilde{\mathcal{K}}$ and label count vectors $\mathbf{m}_i$ and $\mathbf{m}_j$. We next assume that vector $\mathbf{m}_i$ constitutes M_i i.i.d. draws from a categorical distribution with unknown parameters $\boldsymbol{\mu}_i = (\mu_{il})_{l=1}^{|\mathcal{L}|}$, where μ_{il} is the probability that a randomly selected occurrence of raw concept i belongs to an entry in the explanation corpus with label category l. Our label distance d_{label} is the *evidence*

ratio between the count vectors, $\mathbf{m}_i$ and $\mathbf{m}_j$, being drawn from independent categorical distributions (model $\mathcal{M}_{\text{indp}}$) versus them being drawn from the same distribution (model $\mathcal{M}_{\text{comb}}$). More precisely,

$$d_{\text{label}}(\mathbf{m}_i, \mathbf{m}_j) = \frac{p(\mathbf{m}_i, \mathbf{m}_j|\mathcal{M}_{\text{indp}})}{p(\mathbf{m}_i, \mathbf{m}_j|\mathcal{M}_{\text{comb}})}$$

Note that this is not a true distance between count vectors as two identical count vectors do not have a distance of zero. Nonetheless, it satisfies the other requirements of a metric: non-negativity, symmetry and the triangle inequality, and two vectors that are more (less) likely to come from the same multinomial will have a distance less (more) than 1.

To evaluate the label distance we must calculate the evidence for various categorical samples given the model $p(\mathbf{m}|\boldsymbol{\mu}, M)$. For simplicity, we assume total count M is known and define a Dirichlet prior $p(\boldsymbol{\mu}|\alpha\mathbf{1})$ where $\mathbf{1}$ is the vector of all 1s (this makes the simplifying assumption that the prior is symmetric). The evidence for $\mathbf{m}$ is then:

$$p(\mathbf{m}|\alpha) = \int p(\mathbf{m}|\boldsymbol{\mu}, M)p(\boldsymbol{\mu}|\alpha\mathbf{1})d\boldsymbol{\mu}$$

The label meta-metric is the evidence ratio given by:

$$d_{\text{label}}(\mathbf{m}_i, \mathbf{m}_j) = \frac{p(\mathbf{m}_i|\alpha)p(\mathbf{m}_j|\alpha)}{p(\mathbf{m}_i + \mathbf{m}_j|\alpha)}$$

For computational efficiency (and since it did not appear to affect results measurably) we use an approximation for $p(\mathbf{m}|\alpha)$ in our calculation of d_l.

We first evaluate the expected parameter of the posterior distribution given count vector $\mathbf{m}$, namely

$$\tilde{\boldsymbol{\mu}} = \mathbb{E}[\boldsymbol{\mu}|\alpha, \mathbf{m}]$$

then evaluate the evidence for $\mathbf{m}$ conditioned on $\tilde{\boldsymbol{\mu}}$, i.e.

$$p(\mathbf{m}|\tilde{\boldsymbol{\mu}}) = \prod_{k=1}^{K} \left(\frac{m_{il} + \alpha}{M_i + K\alpha} \right)^{m_{il}}$$

We then calculate $\log d_{\text{label}}(\mathbf{m}_i, \mathbf{m}_j)$ and exponentiate to improve precision. After grouping, the raw concept within a cluster with highest frequency is identified as the representative concept of that cluster.

A.3 Language Models

A.3.1 Concept Extraction

After obtaining the free form textual explanations for both datasets, we first cleaned them by removing explanations associated with corrupted video files and the videos which were labelled incorrectly. We then considered three different Spacy's pretrained constituency parsers: *en_core_web_lg*, *en_core_web_md*, *en_core_web_sm* to parse the explanations and extract raw concepts based on the rules discussed in section 5.3.1. We found that, the parser *en_core_web_lg* was more accurate in identifying the constituents and resulted in better concept extraction.

A.3.2 Concept Grouping

For text distance we embedded the raw concepts with a sentence encoder and we experimented with two models: `paraphrase-distilroberta-base-v1` (*distil*) [SDC19] and `stsb-roberta-base` (*stsb*) [RG19] into a 768 dimensional space both using the `sentence_transformer` python library.

And to cluster the semantically similar concepts together using agglomerative clustering [Mul11], we evaluated a variety of distance metrics within the resulting 768-dimensional space, including: Our proposed meta-distance metric, Chebyshev (infinity norm), manhattan, Euclidean and cosine distances. To select hyperparameters, including prior α for label distance, and relative importance factor λ we performed a grid search and selected the values that resulted in well-formed clusters. Based on our experiments (Provided in code), we found that *stsb* encoder with our proposed meta-distance metric resulted in the best grouping of concepts. Note: d_{text} can be either cosine or manhattan distance as they gave similar clusters.

After clustering, we set the frequency occurrence threshold of 3 and removed the rare concept groups which occurred less than this threshold. Then, pruning was done using 90% of mutual information score as discussed in section 5.3.1 which resulted in 78 significant concepts for our MLB-V2E dataset and 62 concepts for MSR-V2E dataset as shown in Table 5.2.

Therefore, each video was associated with a binary concept vector of shape $[1 \times k]$ where k is the number of concepts, indicating the presence and absence of each concept.

A.4 The Classification Models

We considered three different *feature extractors* : Resnet 50v2 [HZR16], Resnet 101v2 [HZR16] and InceptionV3 [SVI16] models and pretrained on the Imagenet dataset to extract features from each frame of our video clips. We excluded the final classification layer from these models and did a global maxpool across the width and height such that we get a 2048 size feature vector for every frame. We then concatenate the features together, resulting in a $[2048 \times 360]$ feature matrix for every video where 360 is the number of frames per video.

For the *Temporal Layer*, we considered both temporal convolution [LFV17] and LSTM [HS97] based architectures which are good at extracting temporal features and found that Temporal CNNs outperformed LSTM by a significant amount. And the *Bottleneck Layer* is a dense layer with k neurons and hence the output is a vector of shape $[1 \times k]$ where k is the number of significant concepts. We introduced an attention layer in the concept-bottleneck model that gives the concept score for each concept. The final fully connected layer is implemented with L neurons (L classes) which predicts the class from the video.

A.5 Performance of Models

Table A.1 and Table A.2 show the performance of all the models with different feature extractors on the two datasets. Each model was trained thrice and the mean and standard deviation of the performance are reported. We find that models with Inception V3 as the feature extractor performed the best. Adding attention mechanism improved the performance of concepts prediction and also achieved higher accuracies than the concept bottleneck models without attention.

Feature Extractor	Model Type	Task Classification		Concepts
		Accuracy(%)	F1-score	AUC
Resnet 50V2	Standard	67.92 ± 0.78	0.68 ± 0.003	-
	CB	67.83 ± 0.74	0.68 ± 0.001	0.85 ± 0.005
	CB + Attn.	67.96 ± 0.65	0.68 ± 0.002	0.88 ± 0.002
Resnet 101V2	Standard	68.18 ± 0.88	0.68 ± 0.005	-
	CB	68.01 ± 1.02	0.68 ± 0.013	0.85 ± 0.004
	CB + Attn.	68.26 ± 1.12	0.68 ± 0.009	0.88 ± 0.000
Inception V3	Standard	68.46 ± 1.27	0.68 ± 0.011	-
	CB	68.16 ± 1.12	0.68 ± 0.004	0.85 ± 0.003
	CB + Attn.	68.38 ± 1.34	0.68 ± 0.004	0.88 ± 0.001

Table A.1: Performance of Models on the MLB-V2E Dataset

Feature Extractor	Model Type	Task Classification		Concepts
		Accuracy(%)	F1-score	AUC
Resnet 50V2	Standard	61.52 ± 1.20	0.59 ± 0.008	-
	CB	61.23 ± 1.71	0.59 ± 0.008	0.82 ± 0.009
	CB + Attn	61.28 ± 1.54	0.59 ± 0.007	0.86 ± 0.004
Resnet 101V2	Standard	61.56 ± 1.31	0.60 ± 0.008	-
	CB	61.38 ± 1.24	0.60 ± 0.008	0.83 ± 0.006
	CB + Attn.	61.44 ± 1.32	0.60 ± 0.009	0.86 ± 0.003
Inception V3	Standard	61.79 ± 1.42	0.60 ± 0.012	-
	CB	61.42 ± 1.18	0.60 ± 0.013	0.83 ± 0.006
	CB + Attn.	61.68 ± 1.23	0.60 ± 0.009	0.86 ± 0.004

Table A.2: Performance of Models on the MSR-V2E Dataset

REFERENCES

[1T20] "One trillion new IoT devices will be produced by 2035." `https://learn.arm.com/route-to-trillion-devices.html`, 2020.

[AA19] Al-Maamoon R Abdali and Rana F Al-Tuma. "Robust real-time violence detection in video using cnn and lstm." In *2019 2nd Scientific Conference of Computer Sciences (SCCS)*, pp. 104–108. IEEE, 2019.

[AB10] Kerem Altun and Billur Barshan. "Human activity recognition using inertial/magnetic sensor units." In *International workshop on human behavior understanding*, pp. 38–51. Springer, 2010.

[ABT20] Abdourrahmane M Atto, Rosie R Bisset, and Emmanuel Trouvé. "Frames learned by prime convolution layers in a deep learning framework." *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[ACA19] Poojitha Amin, Ludmila Cherkasova, Rob Aitken, and Vikas Kache. "Analysis and Demand Forecasting of Residential Energy Consumption at Multiple Time Scales." In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 494–499. IEEE, 2019.

[ACG16] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. "Deep learning with differential privacy." In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, 2016.

[AGO12] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine." In *International workshop on ambient assisted living*, pp. 216–223. Springer, 2012.

[ALB18] Mohammad ABS Akhonda, Yuri Levin-Schwartz, Suchita Bhinge, Vince D Calhoun, and Tülay Adali. "Consecutive independence and correlation transform for multimodal fusion: Application to EEG and fMRI data." In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2311–2315. IEEE, 2018.

[ALK10] Louis Atallah, Benny Lo, Rachel King, and Guang-Zhong Yang. "Sensor placement for activity detection using wearable accelerometers." In *2010 International Conference on Body Sensor Networks*, pp. 24–29. IEEE, 2010.

[Amf10] OD Amft. "On the need for quality standards in activity recognition using ubiquitous sensors." In *Workshop in conjunction with pervasive 2010, how to do good research In activity recognition*, 2010.

[AS19] Roy Assaf and Anika Schumann. "Explainable Deep Neural Networks for Multivariate Time Series Predictions." In *IJCAI*, pp. 6488–6490, 2019.

[ASS19] Sayma Akther, Nazir Saleheen, Shahin Alan Samiei, Vivek Shetty, Emre Ertin, and Santosh Kumar. "mORAL: An mHealth Model for Inferring Oral Hygiene Behaviors in-the-wild Using Wrist-worn Inertial Sensors." *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, **3**(1):1, 2019.

[AW96] Martin F Arlitt and Carey L Williamson. "Web server workload characterization: The search for invariants." *ACM SIGMETRICS Performance Evaluation Review*, **24**(1):126–137, 1996.

[AZS14] Shayan Modiri Assari, Amir Roshan Zamir, and Mubarak Shah. "Video classification using semantic concept co-occurrences." In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2529–2536. IEEE, 2014.

[BBF13] Henrik Blunck, Niels Olof Bouvin, Tobias Franke, Kaj Grønbæk, Mikkel B Kjaergaard, Paul Lukowicz, and Markus Wüstenberg. "On heterogeneity in mobile sensing applications aiming at representative data collection." In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pp. 1087–1098. ACM, 2013.

[BCB14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473*, 2014.

[BCR97] José Manuel Benítez, Juan Luis Castro, and Ignacio Requena. "Are artificial neural networks black boxes?" *IEEE Transactions on neural networks*, **8**(5):1156–1164, 1997.

[BKB14] Hans-Peter Brückner, Benjamin Krüger, and Holger Blume. "Reliable orientation estimation for mobile motion capturing in medical rehabilitation sessions based on inertial measurement units." *Microelectronics Journal*, **45**(12):1603–1611, 2014.

[BSL03] Helene Brashear, Thad Starner, Paul Lukowicz, and Holger Junker. "Using multiple sensors for mobile sign language recognition." Georgia Institute of Technology, 2003.

[BWG12] Andreas Bulling, Jamie A Ward, and Hans Gellersen. "Multimodal recognition of reading activity in transit using body-worn sensors." *ACM Transactions on Applied Perception (TAP)*, **9**(1):2, 2012.

[BWW19] Brett K Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P Bhavnani, James Brian Byrd, and Casey S Greene. "Privacy-preserving generative deep neural networks support clinical data sharing." *Circulation: Cardiovascular Quality and Outcomes*, **12**(7):e005122, 2019.

[CFK13] Diane Cook, Kyle D Feuz, and Narayanan C Krishnan. "Transfer learning for activity recognition: A survey." *Knowledge and information systems*, **36**(3):537–556, 2013.

[CG04] Ludmila Cherkasova and Minaxi Gupta. "Analysis of enterprise media server workloads: access patterns, locality, content evolution, and rates of change." *IEEE/ACM Transactions on networking*, **12**(5):781–794, 2004.

[CGW09] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. "Reading tea leaves: How humans interpret topic models." In *Advances in neural information processing systems*, pp. 288–296, 2009.

[Chi13] Peoples R China. "Electricity consumption prediction based on data mining techniques with particle swarm optimization." *International Journal of Database Theory and Application*, **6**(5):153–164, 2013.

[CJ21] Shaoxiang Chen and Yu-Gang Jiang. "Towards Bridging Event Captioner and Sentence Localizer for Weakly Supervised Dense Event Captioning." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8425–8435, 2021.

[CLC13] Zhenyu Chen, Mu Lin, Fanglin Chen, Nicholas D Lane, Giuseppe Cardone, Rui Wang, Tianxing Li, Yiqiang Chen, Tanzeem Choudhury, and Andrew T Campbell. "Unobtrusive sleep monitoring using smartphones." In *Proceedings of the 7th International Conference on Pervasive Computing Technologies for Healthcare*, pp. 145–152. ICST (Institute for Computer Sciences, Social-Informatics and ..., 2013.

[CLP21] Ling Chen, Xiaoze Liu, Liangying Peng, and Menghan Wu. "Deep learning based multimodal complex human activity recognition using wearable devices." *Applied Intelligence*, **51**(6):4029–4042, 2021.

[CLT19] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. "This looks like that: deep learning for interpretable image recognition." In *Advances in neural information processing systems*, pp. 8930–8941, 2019.

[CPC19] Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. "Machine learning interpretability: A survey on methods and metrics." *Electronics*, **8**(8):832, 2019.

[CPR11] Pierluigi Casale, Oriol Pujol, and Petia Radeva. "Human activity recognition from accelerometer data using a wearable device." In *Iberian Conference on Pattern Recognition and Image Analysis*, pp. 289–296. Springer, 2011.

[CSC13] Ricardo Chavarriaga, Hesam Sagha, Alberto Calatroni, Sundara Tejaswi Digumarti, Gerhard Tröster, José del R Millán, and Daniel Roggen. "The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition." *Pattern Recognition Letters*, **34**(15):2033–2042, 2013.

[CSC19] Minhao Cheng, Simranjit Singh, Patrick Chen, Pin-Yu Chen, Sijia Liu, and Cho-Jui Hsieh. "Sign-opt: A query-efficient hard-label adversarial attack." *arXiv preprint arXiv:1909.10773*, 2019.

[CSH18a] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks." In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 839–847. IEEE, 2018.

[CSH18b] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks." In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 839–847. IEEE, 2018.

[CTR17] Supriyo Chakraborty, Richard Tomsett, Ramya Raghavendra, Daniel Harborne, Moustafa Alzantot, Federico Cerutti, Mani Srivastava, Alun Preece, Simon Julier, Raghuveer M Rao, et al. "Interpretability of deep learning models: a survey of results." In *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pp. 1–6. IEEE, 2017.

[CVN20] Jie Chen, Joel Vaughan, Vijay Nair, and Agus Sudjianto. "Adaptive Explainable Neural Networks (AxNNs)." *Available at SSRN 3569318*, 2020.

[CW17] Nicholas Carlini and David Wagner. "Adversarial examples are not easily detected: Bypassing ten detection methods." In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pp. 3–14, 2017.

[CWH19] Yiqiang Chen, Jindong Wang, Meiyu Huang, and Han Yu. "Cross-position activity recognition with stratified transfer learning." *Pervasive and Mobile Computing*, **57**:1–13, 2019.

[CXQ08] Honglong Chang, Liang Xue, Wei Qin, Guangmin Yuan, and Weizheng Yuan. "An integrated MEMS gyroscope array with higher accuracy output." *Sensors*, **8**(4):2886–2899, 2008.

[CZZ16] Yuwen Chen, Kunhua Zhong, Ju Zhang, Qilong Sun, and Xueliang Zhao. "Lstm networks for mobile human activity recognition." In *2016 International Conference on Artificial Intelligence: Technologies and Applications*. Atlantis Press, 2016.

[Dar09] Waltenegus Dargie. "Analysis of time and frequency domain features of accelerometer measurements." In *2009 Proceedings of 18th International Conference on Computer Communications and Networks*, pp. 1–6. IEEE, 2009.

[DCL19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.", 2019.

[DDS09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

[DE96] Thomas J DiCiccio and Bradley Efron. "Bootstrap confidence intervals." *Statistical science*, pp. 189–212, 1996.

[Del18] Deloitte. "Global mobile consumer survey: US edition." `https://www2.deloitte.com/us/en/pages/technology-media-and-telecommunications/articles/global-mobile-consumer-survey-us-edition.html`, 2018. [Online; accessed 09-April-2019].

[DK17] Finale Doshi-Velez and Been Kim. "Towards a rigorous science of interpretable machine learning." *arXiv preprint arXiv:1702.08608*, 2017.

[DSW14] Yujie Dong, Jenna Scisco, Mike Wilson, Eric Muth, and Adam Hoover. "Detecting periods of eating during free-living by tracking wrist motion." *IEEE journal of biomedical and health informatics*, **18**(4):1253–1260, 2014.

[EAA16] Ç Berke Erdaş, Işıl Atasoy, Koray Açıcı, and Hasan Oğul. "Integrating features for accelerometer-based activity recognition." *Procedia Computer Science*, **98**:522–527, 2016.

[EKS96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In *Kdd*, volume 96, pp. 226–231, 1996.

[FH04] Pedro F Felzenszwalb and Daniel P Huttenlocher. "Efficient graph-based image segmentation." *International journal of computer vision*, **59**(2):167–181, 2004.

[Fin11] Steven Finlay. "Multiple classifier architectures and their application to credit risk assessment." *European Journal of Operational Research*, **210**(2):368–378, 2011.

[FLG07] Jianping Fan, Hangzai Luo, Yuli Gao, and Ramesh Jain. "Incorporating concept ontology for hierarchical video classification, annotation, and visualization." *IEEE Transactions on Multimedia*, **9**(5):939–957, 2007.

[FLX04] Jianping Fan, Hangzai Luo, Jing Xiao, and Lide Wu. "Semantic video classification and feature subset selection under context and concept uncertainty." In *Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries, 2004.*, pp. 192–201. IEEE, 2004.

[FPB08] Elisabetta Farella, Augusto Pieracci, Luca Benini, Laura Rocchi, and Andrea Acquaviva. "Interfacing human and computer with wireless body area sensor networks: the WiMoCA solution." *Multimedia Tools and Applications*, **38**(3):337–363, 2008.

[FV17] Ruth C Fong and Andrea Vedaldi. "Interpretable explanations of black boxes by meaningful perturbation." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3429–3437, 2017.

[GBH09] Alec Go, Richa Bhayani, and Lei Huang. "Twitter sentiment classification using distant supervision." *CS224N project report, Stanford*, **1**(12):2009, 2009.

[GBY18] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal. "Explaining Explanations: An Overview of Interpretability of Machine Learning." In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 80–89, 2018.

[gdp18a] "Article 15 EU GDPR "Right of access by the data subject"." `https://www.privacy-regulation.eu/en/article-15-right-of-access-by-the-data-subject-GDPR.htm`, 2018. Online; accessed 04-March-2022.

[gdp18b] "Recital 71 EU GDPR." `https://www.privacy-regulation.eu/en/r71.htm`, 2018. Online; accessed 27-May-2020.

[gdp18c] "Recital 71 EU GDPR." `https://www.privacy-regulation.eu/en/r71.htm`, 2018. Online; accessed 04-March-2022.

[Ger17] Carola Gerwig. "Short term load forecasting for residential buildings—an extensive literature review." In *International Conference on Intelligent Decision Technologies*, pp. 181–193. Springer, 2017.

[GG16] Yarin Gal and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." In *international conference on machine learning*, pp. 1050–1059, 2016.

[GGG19] Alan H Gee, Diego Garcia-Olano, Joydeep Ghosh, and David Paydarfar. "Explaining deep classification of time-series data with learned prototypes." *arXiv preprint arXiv:1904.08935*, 2019.

[GGZ17] Lianli Gao, Zhao Guo, Hanwang Zhang, Xing Xu, and Heng Tao Shen. "Video captioning with attention-based LSTM and semantic consistency." *IEEE Transactions on Multimedia*, **19**(9):2045–2055, 2017.

[GL14] Yoav Goldberg and Omer Levy. "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method." *arXiv preprint arXiv:1402.3722*, 2014.

[GMP17] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. "On the (statistical) detection of adversarial examples." *arXiv preprint arXiv:1702.06280*, 2017.

[GSC99] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM." 1999.

[GWS11] Daniel Göhring, Miao Wang, Michael Schnürmacher, and Tinosch Ganjineh. "Radar/lidar sensor fusion for car-following on highways." In *The 5th International Conference on Automation, Robotics and Applications*, pp. 407–412. IEEE, 2011.

[GWZ19] Amirata Ghorbani, James Wexler, James Zou, and Been Kim. "Towards automatic concept-based explanations." *arXiv preprint arXiv:1902.03129*, 2019.

[HB20] Peter Hase and Mohit Bansal. "Evaluating Explainable AI: Which Algorithmic Explanations Help Users Predict Model Behavior?" *arXiv preprint arXiv:2005.01831*, 2020.

[HCL19] Peter Hase, Chaofan Chen, Oscar Li, and Cynthia Rudin. "Interpretable Image Recognition with Hierarchical Prototypes." In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 7, pp. 32–40, 2019.

[Hec18] Jeff Hecht. "Lidar for self-driving cars." *Optics and Photonics News*, **29**(1):26–33, 2018.

[him18] "Arm Expands IoT Connectivity and Device Management Capabilities with Stream Technologies Acquisition." `https://www.arm.com/company/news/2018/06/arm-expands-iot-connectivity-and-device-management-capabilities-with-stream-` 2018.

[HNT13] Samuli Hemminki, Petteri Nurmi, and Sasu Tarkoma. "Accelerometer-based transportation mode detection on smartphones." In *Proceedings of the 11th ACM conference on embedded networked sensor systems*, p. 13. ACM, 2013.

[HPH20] Liam Hiley, Alun Preece, Yulia Hicks, Supriyo Chakraborty, Prudhvi Gurram, and Richard Tomsett. "Explaining motion relevance for activity recognition in video deep learning models." *arXiv preprint arXiv:2003.14285*, 2020.

[HRH02] Peter D Hoff, Adrian E Raftery, and Mark S Handcock. "Latent space approaches to social network analysis." *Journal of the american Statistical association*, **97**(460):1090–1098, 2002.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." *Neural computation*, **9**(8):1735–1780, 1997.

[HS06] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks." *science*, **313**(5786):504–507, 2006.

[HTL13] Boye A Høverstad, Axel Tidemann, and Helge Langseth. "Effects of data cleansing on load prediction algorithms." In *2013 IEEE Computational Intelligence Applications in Smart Grid (CIASG)*, pp. 93–100. IEEE, 2013.

[Hu17] Yi-Chung Hu. "Electricity consumption prediction using a neural-network-based grey forecasting approach." *Journal of the Operational Research Society*, **68**(10):1259–1264, 2017.

[HZL16] Xue-Qin Huo, Wei-Long Zheng, and Bao-Liang Lu. "Driving fatigue detection with fusion of EEG and forehead EOG." In *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 897–904. IEEE, 2016.

[HZR16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[IDC18] IDC. "New Wearables Forecast from IDC Shows Smartwatches Continuing Their Ascendance While Wristbands Face Flat Growth." `https://www.idc.com/getdoc.jsp?containerId=prUS44000018fdeep`, 2018. [Online; accessed 09-April-2019].

[IS15] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *arXiv preprint arXiv:1502.03167*, 2015.

[ITU15] Takuya Isho, Hideyuki Tashiro, and Shigeru Usuda. "Accelerometry-based gait characteristics evaluated using a smartphone and their association with fall risk in people with chronic stroke." *Journal of stroke and cerebrovascular diseases*, **24**(6):1305–1311, 2015.

[JCL20] Jeya Vikranth Jeyakumar, Ludmila Cherkasova, Saina Lajevardi, Moray Allan, Yue Zhao, John Fry, and Mani Srivastava. "Combining Individual and Joint Networking Behavior for Intelligent IoT Analytics." In *International Conference on Internet of Things*, pp. 45–62. Springer, 2020.

[JDC21] Jeya Vikranth Jeyakumar, Luke Dickens, Yu-Hsi Cheng, Joseph Noor, Luis Antonio Garcia, Diego Ramirez Echavarria, Alessandra Russo, Lance M Kaplan, and Mani Srivastava. "Automatic Concept Extraction for Concept Bottleneck-based Video Classification." 2021.

[JLS19] Jeya Vikranth Jeyakumar, Liangzhen Lai, Naveen Suda, and Mani Srivastava. "SenseHAR: a robust virtual activity sensor for smartphones and wearables." In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, pp. 15–28, 2019.

[JLX18] Jeya Vikranth Jeyakumar, Eun Sun Lee, Zhengxu Xia, Sandeep Singh Sandha, Nathan Tausik, and Mani Srivastava. "Deep convolutional bidirectional LSTM based transportation mode recognition." In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, pp. 1606–1615. ACM, 2018.

[JNC20] Jeya Vikranth Jeyakumar, Joseph Noor, Yu-Hsi Cheng, Luis Garcia, and Mani Srivastava. "How Can I Explain This to You? An Empirical Study of Deep Neural Network Explanation Methods." *Advances in Neural Information Processing Systems*, **33**, 2020.

[KB14] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*, 2014.

[KBB09] Neeraj Kumar, Alexander C Berg, Peter N Belhumeur, and Shree K Nayar. "Attribute and simile classifiers for face verification." In *2009 IEEE 12th international conference on computer vision*, pp. 365–372. IEEE, 2009.

[KGB14] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. "A convolutional neural network for modelling sentences." *arXiv preprint arXiv:1404.2188*, 2014.

[KH09] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images." 2009.

[Kim01] Won Kim. "Characterizing the scalability of a large web-based shopping system." *ACM Transactions on Internet Technology (TOIT)*, **1**(1):44–69, 2001.

[KKK16] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. "Examples are not enough, learn to criticize! criticism for interpretability." In *Advances in neural information processing systems*, pp. 2280–2288, 2016.

[KL17] Pang Wei Koh and Percy Liang. "Understanding black-box predictions via influence functions." In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1885–1894. JMLR. org, 2017.

[KNT20] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. "Concept bottleneck models." In *International Conference on Machine Learning*, pp. 5338–5348. PMLR, 2020.

[KP05] Ashish Kapoor and Rosalind W Picard. "Multimodal affect recognition in learning environments." In *Proceedings of the 13th annual ACM international conference on Multimedia*, pp. 677–682. ACM, 2005.

[KRM18] Md Abdullah Al Hafiz Khan, Nirmalya Roy, and Archan Misra. "Scaling human activity recognition via deep learning-based domain adaptation." In *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–9. IEEE, 2018.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[KVV19] Janis Klaise, Arnaud Van Looveren, Giovanni Vacanti, and Alexandru Coca. "Alibi: Algorithms for monitoring and explaining machine learning models.", 2019.

[KWG18] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)." In *International conference on machine learning*, pp. 2668–2677. PMLR, 2018.

[LAT19] Paula Lago, Sayeda Shamma Alia, Shingo Takeda, Tittaya Mairittha, Nattaya Mairittha, Farina Faiz, Yusuke Nishimura, Kohei Adachi, Tsuyoshi Okita, François Charpillet, et al. "Nurse care activity recognition challenge: summary and results." In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, pp. 746–751, 2019.

[LC11] Young-Seol Lee and Sung-Bae Cho. "Activity recognition using hierarchical hidden markov models on a smartphone with 3D accelerometer." In *International Conference on Hybrid Artificial Intelligence Systems*, pp. 460–467. Springer, 2011.

[LFV17] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. "Temporal convolutional networks for action segmentation and detection." In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 156–165, 2017.

[LGM20] Q Vera Liao, Daniel Gruen, and Sarah Miller. "Questioning the AI: informing design practices for explainable AI user experiences." In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–15, 2020.

[LJB17] Eun Sun Lee, Jeya Vikranth Jeyakumar, Bharathan Balaji, Rory P Wilson, and Mani Srivastava. "AquaMote: Ultra low power sensor tag for animal localization and fine motion tracking." In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, pp. 1–2, 2017.

[LL04] Xie Lingyun and Du Limin. "Efficient Viterbi beam search algorithm using dynamic pruning." In *Proceedings 7th International Conference on Signal Processing, 2004. Proceedings. ICSP'04. 2004.*, volume 1, pp. 699–702. IEEE, 2004.

[LL17] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions." In *Advances in neural information processing systems*, pp. 4765–4774, 2017.

[LLL13] Robert LiKamWa, Yunxin Liu, Nicholas D. Lane, and Lin Zhong. "MoodScope: Building a Mood Sensor from Smartphone Usage Patterns." In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '13, pp. 389–402, New York, NY, USA, 2013. ACM.

[LML10] Nicholas D Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T Campbell. "A survey of mobile phone sensing." *IEEE Communications magazine*, **48**(9):140–150, 2010.

[LQX20] Jonathan Lam, Pengrui Quan, Jiamin Xu, Jeya Vikranth Jeyakumar, and Mani Srivastava. "Hard-Label Black-Box Adversarial Attack on Deep Electrocardiogram Classifier." In *Proceedings of the 1st ACM International Workshop on Security and Safety for Intelligent Cyber-Physical Systems*, pp. 6–12, 2020.

[LSH09] Qiang Li, John A Stankovic, Mark A Hanson, Adam T Barth, John Lach, Gang Zhou, et al. "Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information." In *BSN*, volume 9, pp. 138–143, 2009.

[LSP06] Michael P Linegang, Heather A Stoner, Michael J Patterson, Bobbie D Seppelt, Joshua D Hoffman, Zachariah B Crittendon, and John D Lee. "Human-automation collaboration in dynamic mission planning: A challenge requiring an ecological approach." In *Proceedings of the human factors and ergonomics society annual meeting*, volume 50, pp. 2482–2486. SAGE Publications Sage CA: Los Angeles, CA, 2006.

[LT19] Piyawat Lertvittayakumjorn and Francesca Toni. "Human-grounded evaluations of explanation methods for text classification." *arXiv preprint arXiv:1908.11355*, 2019.

[LWJ04] Paul Lukowicz, Jamie A Ward, Holger Junker, Mathias Stäger, Gerhard Tröster, Amin Atrash, and Thad Starner. "Recognizing workshop activity using body worn microphones and accelerometers." In *International conference on pervasive computing*, pp. 18–32. Springer, 2004.

[Mac03] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[mas16] "SoftBank CEO Masayoshi Son sees a future with 1 trillion Internet of Things devices." `https://venturebeat.com/2016/10/25/softbank-ceo-masayoshi-son-sees-a-future-with-1-trillion-internet-of-things` 2016.

[MGF17] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. "On detecting adversarial perturbations." *arXiv preprint arXiv:1702.04267*, 2017.

[MHL15] Fen Miao, Yi He, Jinlei Liu, Ye Li, and Idowu Ayoola. "Identifying typical physical activity on smartphone with varying positions and orientations." *Biomedical engineering online*, **14**(1):32, 2015.

[Mil19] Tim Miller. "Explanation in artificial intelligence: Insights from the social sciences." *Artificial intelligence*, **267**:1–38, 2019.

[MKL20] Shikhar Murty, Pang Wei Koh, and Percy Liang. "Expbert: Representation engineering with natural language explanations." *arXiv preprint arXiv:2005.01932*, 2020.

[MM01] George B Moody and Roger G Mark. "The impact of the MIT-BIH arrhythmia database." *IEEE Engineering in Medicine and Biology Magazine*, **20**(3):45–50, 2001.

[MMC11] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. "Stacked convolutional auto-encoders for hierarchical feature extraction." In *International conference on artificial neural networks*, pp. 52–59. Springer, 2011.

[MOF19] Marc Moreaux, Michael Garcia Ortiz, Isabelle Ferrané, and Frédéric Lerasle. "Benchmark for kitchen20, a daily life dataset for audio-based human action recognition." In *2019 International Conference on Content-Based Multimedia Indexing (CBMI)*, pp. 1–6. IEEE, 2019.

[MR16] Francisco Javier Ordóñez Morales and Daniel Roggen. "Deep convolutional feature transfer across mobile activity recognition domains, sensor modalities and locations." In *Proceedings of the 2016 ACM International Symposium on Wearable Computers*, pp. 92–99. ACM, 2016.

[MRC16] Joseph E Mercado, Michael A Rupp, Jessie YC Chen, Michael J Barnes, Daniel Barber, and Katelyn Procci. "Intelligent agent transparency in human–agent teaming for Multi-UxV management." *Human factors*, **58**(3):401–415, 2016.

[Mul11] Daniel Müllner. "Modern hierarchical, agglomerative clustering algorithms." *arXiv preprint arXiv:1109.2378*, 2011.

[NDE12] Vincenzo Natale, Maciek Drejak, Alex Erbacci, Lorenzo Tonetti, Marco Fabbri, and Monica Martoni. "Monitoring sleep with a smartphone accelerometer." *Sleep and Biological Rhythms*, **10**(4):287–292, 2012.

[NKK11] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. "Multimodal deep learning." In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 689–696, 2011.

[NMN15] Trung Thanh Ngo, Yasushi Makihara, Hajime Nagahara, Yasuhiro Mukaigawa, and Yasushi Yagi. "Similar gait action recognition using an inertial sensor." *Pattern Recognition*, **48**(4):1289–1301, 2015.

[NR15] Domen Novak and Robert Riener. "A survey of sensor fusion methods in wearable robotics." *Robotics and Autonomous Systems*, **73**:155–170, 2015.

[NSF17] Cristina Nichiforov, Iulia Stamatescu, Ioana Făgărăşan, and Grigore Stamatescu. "Energy consumption forecasting using ARIMA and neural network models." In *2017 5th International Symposium on Electrical and Electronics Engineering (ISEEE)*, pp. 1–4. IEEE, 2017.

[NW02] Joanna Nowicka-Zagrajek and Rafal Weron. "Modeling electricity loads in California: ARMA models with hyperbolic noise." *Signal Processing*, **82**(12):1903–1915, 2002.

[NWF78] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. "An analysis of approximations for maximizing submodular set functions I." *Mathematical Programming*, **14**:265–294, 1978.

[PCC14] Abhinav Parate, Meng-Chieh Chiu, Chaniel Chadowitz, Deepak Ganesan, and Evangelos Kalogerakis. "Risq: Recognizing smoking gestures with inertial sensors on a wristband." In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pp. 149–161. ACM, 2014.

[PCY18] Liangying Peng, Ling Chen, Zhenan Ye, and Yi Zhang. "Aroma: A deep multi-task learning based simple and complex human activity recognition method using wearable sensors." *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, **2**(2):1–16, 2018.

[PDM12] Slav Petrov, Dipanjan Das, and Ryan McDonald. "A Universal Part-of-Speech Tagset." In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pp. 2089–2096, 2012.

[PDS18] Vitali Petsiuk, Abir Das, and Kate Saenko. "Rise: Randomized input sampling for explanation of black-box models." *arXiv preprint arXiv:1806.07421*, 2018.

[PEA07] Juha Parkka, Miikka Ermes, Kari Antila, Mark Van Gils, Ari Manttari, and Heikki Nieminen. "Estimating intensity of physical activity: a comparison of wearable accelerometer and gyro sensors and 3 sensor locations." In *2007 29th annual international conference of the IEEE engineering in medicine and biology society*, pp. 1511–1514. IEEE, 2007.

[PEA10] Giuseppe Paleologo, André Elisseeff, and Gianluca Antonini. "Subagging for credit scoring models." *European Journal of Operational Research*, **201**(2):490–499, 2010.

[pel20a] "Arm Pelion Device Management - Deploying and managing IoT devices at large scale." `https://www.pelion.com/iot-device-management/`, 2020.

[pel20b] "Arm Pelion IoT Platform: Connectivity Management." `https://www.arm.com/products/iot/pelion-iot-platform/connectivity-management`, 2020.

[Pic] Karol J. Piczak. "ESC: Dataset for Environmental Sound Classification." In *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pp. 1015–1018. ACM Press.

[PM18] Nicolas Papernot and Patrick McDaniel. "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning." *arXiv preprint arXiv:1803.04765*, 2018.

[pow] "Power law — Wikipedia, The Free Encyclopedia." `https://en.wikipedia.org/wiki/Power_law`. [Online; accessed 25-April-2020].

[PR18] AJ Piergiovanni and Michael S. Ryoo. "Fine-grained Activity Recognition in Baseball Videos." In *CVPR Workshop on Computer Vision in Sports*, 2018.

[pro20] "Interpretable Image Classification with Prototypes." `https://aihub.cloud.google.com/u/0/p/products%2F78e67336-74a0-4a63-b367-7b395567ffba`, 2020. Online; accessed 27-May-2020.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[PYL17] Yingwei Pan, Ting Yao, Houqiang Li, and Tao Mei. "Video Captioning With Transferred Semantic Attributes." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[RCR10] Daniel Roggen, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Förster, Gerhard Tröster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, et al. "Collecting complex activity datasets in highly rich networked sensor environments." In *2010 Seventh international conference on networked sensing systems (INSS)*, pp. 233–240. IEEE, 2010.

[Rea19] S Reardon. “Rise of Robot Radiologists.” *Nature*, **576**(7787):S54, 2019.

[RG18] Seyed Ali Rokni and Hassan Ghasemzadeh. “Autonomous training of activity recognition algorithms in mobile sensors: A transfer learning approach in context-invariant views.” *IEEE Transactions on Mobile Computing*, **17**(8):1764–1777, 2018.

[RG19] Nils Reimers and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.” In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[RMB10] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. “Using mobile phones to determine transportation modes.” *ACM Transactions on Sensor Networks (TOSN)*, **6**(2):13, 2010.

[RS12a] Attila Reiss and Didier Stricker. “Creating and benchmarking a new dataset for physical activity monitoring.” In *Proceedings of the 5th International Conference on PErvasive Technologies Related to Assistive Environments*, p. 40. ACM, 2012.

[RS12b] Attila Reiss and Didier Stricker. “Introducing a New Benchmarked Dataset for Activity Monitoring.” In *Proceedings of the 2012 16th Annual International Symposium on Wearable Computers (ISWC)*, ISWC ’12, pp. 108–109, Washington, DC, USA, 2012. IEEE Computer Society.

[RSG16] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “" Why should i trust you?" Explaining the predictions of any classifier.” In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

[RSG18] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “Anchors: High-Precision Model-Agnostic Explanations.” In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[RTB18] Valentin Radu, Catherine Tong, Sourav Bhattacharya, Nicholas D Lane, Cecilia Mascolo, Mahesh K Marina, and Fahim Kawsar. “Multimodal deep learning for activity and context recognition.” *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, **1**(4):157, 2018.

[Rud19] Cynthia Rudin. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead.” *Nature Machine Intelligence*, **1**(5):206–215, 2019.

[SAE19] Udo Schlegel, Hiba Arnout, Mennatallah El-Assady, Daniela Oelke, and Daniel A Keim. “Towards a rigorous evaluation of XAI Methods on Time Series.” *arXiv preprint arXiv:1909.07082*, 2019.

[SBB15] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. "Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition." In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pp. 127–140. ACM, 2015.

[SBH05] Kyung-Bin Song, Young-Sik Baek, Dug Hun Hong, and Gilsoo Jang. "Short-term load forecasting for the holidays using fuzzy linear regression method." *IEEE transactions on power systems*, **20**(1):96–101, 2005.

[SBI16] Muhammad Shoaib, Stephan Bosch, Ozlem Incel, Hans Scholten, and Paul Havinga. "Complex human activity recognition using smartphone and wrist-worn motion sensors." *Sensors*, **16**(4):426, 2016.

[SBV11] Christian Seeger, Alejandro Buchmann, and Kristof Van Laerhoven. "myHealthAssistant: a phone-based body sensor network that captures the wearer's exercises throughout the day." In *Proceedings of the 6th International Conference on Body Area Networks*, pp. 1–7. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2011.

[SCD17] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. "Grad-cam: Visual explanations from deep networks via gradient-based localization." In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

[SDC19] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." *ArXiv*, **abs/1910.01108**, 2019.

[SGA17] Fendy Santoso, Matthew A Garratt, and Sreenatha G Anavatti. "Visual–inertial navigation systems for aerial robotics: Sensor fusion and technology." *IEEE Transactions on Automation Science and Engineering*, **14**(1):260–275, 2017.

[SHK14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." *The Journal of Machine Learning Research*, **15**(1):1929–1958, 2014.

[SJD16] SM Sulaiman, P Aruna Jeyanthy, and D Devaraj. "Artificial neural network based day ahead load forecasting using Smart Meter data." In *2016 Biennial International Conference on Power and Energy Systems: Towards Sustainable Energy (PESTSE)*, pp. 1–6. IEEE, 2016.

[SKK14] Jørgen Skotte, Mette Korshøj, Jesper Kristiansen, Christiana Hanisch, and Andreas Holtermann. "Detection of physical activity types using triaxial accelerometers." *Journal of physical activity and health*, **11**(1):76–84, 2014.

[SRS17] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. "Machine learning models that remember too much." In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 587–601, 2017.

[Sta18] Statista. "Global smartphone shipments forecast from 2010 to 2022 (in million units)." `https://www.statista.com/statistics/263441/global-smartphone-shipments-forecast/`, 2018. [Online; accessed 09-April-2019].

[SVI16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[SVZ13] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps." *arXiv preprint arXiv:1312.6034*, 2013.

[TFC07] Wenting Tang, Yun Fu, Ludmila Cherkasova, and Amin Vahdat. "Modeling and generating realistic streaming media server workloads." *Computer Networks*, **51**(1):336–356, 2007.

[TIH07] Emmanuel Munguia Tapia, Stephen S Intille, William Haskell, Kent Larson, Julie Wright, Abby King, and Robert Friedman. "Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor." In *Wearable Computers, 2007 11th IEEE International Symposium on*, pp. 37–40. IEEE, 2007.

[VBD96] Peter H Veltink, HansB J Bussmann, Wiebe De Vries, WimL J Martens, Rob C Van Lummel, et al. "Detection of static and dynamic activities using uniaxial accelerometers." *IEEE Transactions on Rehabilitation Engineering*, **4**(4):375–385, 1996.

[VSB18] Joel Vaughan, Agus Sudjianto, Erind Brahimi, Jie Chen, and Vijayan N Nair. "Explainable neural networks based on additive index models." *arXiv preprint arXiv:1806.01933*, 2018.

[VVL07] Douglas L Vail, Manuela M Veloso, and John D Lafferty. "Conditional random fields for activity recognition." In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, p. 235. ACM, 2007.

[WAP13] Waskitho Wibisono, Dedy Nur Arifin, Baskoro Adi Pratomo, Tohari Ahmad, and Royyana M Ijtihadie. "Falls detection and notification system using tri-axial accelerometer and gyroscope sensors of a smartphone." In *2013 Conference on Technologies and Applications of Artificial Intelligence*, pp. 382–385. IEEE, 2013.

[War18] Pete Warden. "Speech commands: A dataset for limited-vocabulary speech recognition." *arXiv preprint arXiv:1804.03209*, 2018.

[WCM20] Jialin Wu, Liyan Chen, and Raymond J Mooney. "Improving VQA and its Explanations by Comparing Competing Explanations." *arXiv preprint arXiv:2006.15631*, 2020.

[WGM18] Lin Wang, Hristijan Gjoreskia, Kazuya Murao, Tsuyoshi Okita, and Daniel Roggen. "Summary of the sussex-huawei locomotion-transportation recognition challenge." In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, pp. 1521–1530. ACM, 2018.

[wik] "Pareto principle — Wikipedia, The Free Encyclopedia." `https://en.wikipedia.org/wiki/Pareto_principle`. [Online; accessed 25-April-2020].

[WJW16] Zuxuan Wu, Yu-Gang Jiang, Xi Wang, Hao Ye, and Xiangyang Xue. "Multi-stream multi-class fusion of deep networks for video classification." In *Proceedings of the 24th ACM international conference on Multimedia*, pp. 791–800. ACM, 2016.

[WMZ18] Bairui Wang, Lin Ma, Wei Zhang, and Wei Liu. "Reconstruction network for video captioning." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7622–7631, 2018.

[WRD16] Hongyi Wen, Julian Ramos Rojas, and Anind K Dey. "Serendipity: Finger gesture recognition using an off-the-shelf smartwatch." In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 3847–3851. ACM, 2016.

[WSH09] Xiaozhe Wang, Kate Smith-Miles, and Rob Hyndman. "Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series." *Neurocomputing*, **72**(10-12):2581–2594, 2009.

[WZX16] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling." In *Advances in Neural Information Processing Systems*, pp. 82–90, 2016.

[XCW15] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting." In *Advances in neural information processing systems*, pp. 802–810, 2015.

[XMY16] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. "Msr-vtt: A large video description dataset for bridging video and language." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5288–5296, 2016.

[XSB18] Tianwei Xing, Sandeep Singh Sandha, Bharathan Balaji, Supriyo Chakraborty, and Mani Srivastava. "Enabling Edge Devices that Learn from Each Other: Cross Modal Training for Activity Recognition." In *Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking*, pp. 37–42. ACM, 2018.

[YC08] Sung-Ihk Yang and Sung-Bae Cho. "Recognizing human activities from accelerometer and physiological sensors." In *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 100–105. IEEE, 2008.

[YHZ17] Shuochao Yao, Shaohan Hu, Yiran Zhao, Aston Zhang, and Tarek Abdelzaher. "Deepsense: A unified deep learning framework for time-series mobile sensing data processing." In *Proceedings of the 26th International Conference on World Wide Web*, pp. 351–360. International World Wide Web Conferences Steering Committee, 2017.

[YKA20] Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. "On Completeness-aware Concept-Based Explanations in Deep Neural Networks." *Advances in Neural Information Processing Systems*, **33**, 2020.

[YKC17] Youngjae Yu, Hyungjin Ko, Jongwook Choi, and Gunhee Kim. "End-to-end concept word detection for video captioning, retrieval, and question answering." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3165–3173, 2017.

[YLM13] O Yurur, C-H Liu, and W Moreno. "Unsupervised posture detection by smartphone accelerometer." *Electronics Letters*, **49**(8):562–564, 2013.

[YRJ19] Serena Yeung, Francesca Rinaldo, Jeffrey Jopling, Bingbin Liu, Rishab Mehra, N Lance Downing, Michelle Guo, Gabriel M Bianconi, Alexandre Alahi, Julia Lee, et al. "A computer vision system for deep learning-based detection of patient mobilization activities in the ICU." *NPJ digital medicine*, **2**(1):1–5, 2019.

[YTW19] Chenggang Yan, Yunbin Tu, Xingzheng Wang, Yongbing Zhang, Xinhong Hao, Yongdong Zhang, and Qionghai Dai. "STAT: Spatial-temporal attention mechanism for video captioning." *IEEE transactions on multimedia*, **22**(1):229–241, 2019.

[ZNY14] Ming Zeng, Le T Nguyen, Bo Yu, Ole J Mengshoel, Jiang Zhu, Pang Wu, and Joy Zhang. "Convolutional neural networks for human activity recognition using mobile sensors." In *6th International Conference on Mobile Computing, Applications and Services*, pp. 197–205. IEEE, 2014.

[ZPI17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. "Unpaired image-to-image translation using cycle-consistent adversarial networks." *arXiv preprint*, 2017.

[ZS12] Mi Zhang and Alexander A Sawchuk. "USC-HAD: a daily activity dataset for ubiquitous activity recognition using wearable sensors." In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 1036–1043. ACM, 2012.

[ZSF07] Piero Zappi, Thomas Stiefmeier, Elisabetta Farella, Daniel Roggen, Luca Benini, and Gerhard Troster. "Activity recognition from on-body sensors by classifier fusion: sensor scalability and robustness." In *2007 3rd international conference on intelligent sensors, sensor networks and information*, pp. 281–286. IEEE, 2007.

[ZZC18] Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. "End-to-end dense video captioning with masked transformer." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8739–8748, 2018.