Prediction and Observation Timing in Time Series Classification

by

Thomas Hartvigsen

A Dissertation

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in

Data Science

December 2021

APPROVED:

Professor Elke A. Rundensteiner
Worcester Polytechnic Institute
Advisor

Professor Xiangnan Kong
Worcester Polytechnic Institute
Co-Advisor

Professor Randy Paffenroth
Worcester Polytechnic Institute
Committee Member

Professor Jenna Wiens
University of Michigan
External Committee Member

Professor Elke A. Rundensteiner
Worcester Polytechnic Institute
Head of Data Science Program

Abstract

Time series data mining is crucial to a wide variety of domains such as healthcare, weather prediction, seismology, and astronomy. Classifying time series is a challenging and important problem with applications from clinical diagnosis, natural disaster effect estimation, to stellar object detection. With a huge amount of time series data collected every day, solving rapidly-evolving, complex time series classification problems is essential. In this work, we introduce and study two notions of *timing* in time series classification: (1) The timing of *predictions*, or *how early in time a classification is made without seeing the complete time series*, is often a key factor in the *usefulness* of a time series classifier, and (2) The timing of *observations*, or timestamped values produced by various data sources, is often irregular across different data dimensions, thus introducing technical challenges to the classifier model.

In the first part of this dissertation, we study *prediction timing*. How *quickly* a classifier comes to a decision can strongly effect its *usefulness* when integrated into a decision support system. This intuition underlies *early classification of time series*, where we find a timestep in an ongoing time series at which a classifier stops and makes its prediction *early* without having seen the complete time series, typically with a preference for consuming only a few early timesteps and accurate predictions. These two goals of earliness and high accuracy contradict one another. We investigate two directions for tackling prediction timing:

Tunable Early Classification of Time Series. A *tunable* early classifier lets a user choose how much importance to put on each of the contradictory goals of *earliness* and *accuracy*. Since no prior works have tackled this challenge, in this work we characterize propose the tunable early classification problem. We then develop a solution strategy based on a recurrent neural network time series model mixed with a reinforcement learning-based halting policy that chooses, at each timestep, whether or not to stop and classify.

Early Multi-label Classification of Time Series. Many time series classification problems are naturally modeled as the more general case of *multi-label* classification, where each instance is associated with a *subset* of all possible labels. Classifying such series early is an open problem, which we refer to as Early Multi-label Classification. In this dissertation, we develop its first solution: An integration of recent classifier chain approaches with multi-label classification models as well as with adaptive-halting policy networks.

In the second part of this dissertation, we study *observation* timing. Time series are often collected with *irregular* spacing between different observations (data values). Classi-

fying such irregular time series has many important applications from clinical diagnosis to seismology. However, modeling these data is tremendously challenging because sampling rates can differ between variables, values may be missing, variables evolve over time without being observed, and data generation functions may or may not be correlated. In this dissertation, we develop classifiers that learn *directly* from irregular time series, studying two challenges related to directions for observation timing. The first studies the classification of long and irregular time series, while the second addresses data irregularity in the context of achieving early classification.

Continuous-Time Attention for Irregular Time Series Classification. Classifying time series often depends on finding discriminative subsequences in time series data. A model that explicitly finds these *moments-of-interest* in *continuous time* will be more robust to noise by disregarding irrelevant portions the timeline. In this work, we leverage the power of *attention* to find relevant moments-of-interest.

Classifying Irregular Time Series Early. Once we can find moments-of-interest, a natural follow-on question arises, namely, *Can we find them early to support time-sensitive applications?* We investigate this question, developing a model that classifies irregular time series *as early as possible* without first observing all timesteps. This is the first work to combine *observation timing* with *prediction timing*.

For all tasks, we compare our proposed models against state-of-the-art alternatives from the literature and verify that our proposed methods outperform them consistently and significantly. In these studies, we measure performance using established metrics on a wide variety of publicly-available datasets.

Acknowledgments

I am deeply grateful to my advisors Professor Elke Rundensteiner and Professor Xiangnan Kong for their support. I will forever strive to pay forward their investments in time, thoughtful criticism, and drive to future generations of researchers. Their patience and trust have helped me to grow as both researcher and person. Also special thank you to Professor Randy Paffenroth and Professor Jenna Wiens for serving on my dissertation committee.

I also thank my fellow students who have helped me along the way: Cansu Sen, Walter Gerych, Jidapa Thadajarassiri, Thanh Tran, Xiao Qin, John Boaz Lee, and all other members of the DAISY group at WPI. Through many discussions, they have helped me to understand problems more deeply and their hard work inspires me to keep pushing forward. Especially thank you to Cansu for listening to my research ravings and for dodging my flailing arms as I failed to contain my enthusiasm for our fabulous research.

I am especially grateful to my wife, Missy, who has demonstrated extreme patience through periods of high stress and has led the charge in celebrating any and all successes. Finally, thank you to my parents, Gregg and Meredith, and my sister, Phoebe, for their support and for inspiring me to stay curious and motivated.

Publications

Publications featured in this dissertation

This dissertation describes work done in the four following papers. Together, they build solutions for timing and observation prediction problems in time series classification.

21. Thomas Hartvigsen, Walter Gerych, Xiangnan Kong, Elke Rundensteiner. *Early Classification of Irregular Time Series*. Forthcoming.

20. Thomas Hartvigsen, Xiangnan Kong, Elke Rundensteiner. *Continuous-Time Attention Network for Irregularly-Sampled Time Series*. Forthcoming.

19. Thomas Hartvigsen, Cansu Sen, Xiangnan Kong, Elke Rundensteiner. *Recurrent Halting Chain for Early Multi-label Classification.* ACM SIGKDD Conference on Knowledge Discovery and Data Mining (**KDD**), 2020.

18. Thomas Hartvigsen, Cansu Sen, Xiangnan Kong, Elke Rundensteiner. *Adaptive-Halting Policy Network for Early Classification.* ACM SIGKDD Conference on Knowledge Discovery and Data Mining (**KDD**), 2019.

Other Publications

The following papers have also been accepted for publication when this dissertation was submitted. Nine additional papers are currently under review.

17. Walter Gerych, Thomas Hartvigsen, Luke Buquicchio, Emmanuel Agu, Elke Rundensteiner. *Recovering the Propensity Score from Biased Positive Unlabeled Data*. AAAI Conference on Artificial Intelligence (**AAAI**), 2022.

16. Walter Gerych, Thomas Hartvigsen, Luke Buquicchio, Emmanuel Agu, Elke Rundensteiner. *Recurrent Bayesian Classifier Chains for Exact Multi-label Classification*. Advances in Neural Information Processing Systems (**NeurIPS**), 2021.

15. Hang Yin, John Boaz Lee, Xiangnan Kong, Thomas Hartvigsen, Sihong Xie. *Energy-Efficient Models for High-Dimensional Spike Train Classification using Sparse Spiking Neural Networks.* ACM SIGKDD Conference on Knowledge Discovery and Data Mining (**KDD**), 2021.

14. Jidapa Thadajarassiri, Thomas Hartvigsen, Xiangnan Kong, Elke Rundensteiner. *Semi-Supervised Knowledge Amalgamation for Sequence Classification.* AAAI Conference on Artificial Intelligence (**AAAI**), 2021.

13. Luke Buquicchio, Walter Gerych, Kavin Chandrasekaran, Abdulaziz Alajaji, Hamid Mansoor, Thomas Hartvigsen, Emmanuel Agu, and Elke Rundensteiner. *Variational Open-Set Recognition*. IEEE International Conference on Big Data (**IEEE BigData**), 2021.

12. Dongyu Zhang, Cansu Sen, Jidapa Thadajarassiri Thomas Hartvigsen, Xiangnan Kong, Elke Rundensteiner. *Explainable Document Classification with Human-guided Attention*. IEEE International Conference on Big Data (**IEEE BigData**), 2021.

11. Prathyush Parvatharaju, Ramesh Doddiah, Thomas Hartvigsen, Elke Rundensteiner. *Learning Saliency Maps for Deep Time Series Classifiers*. ACM International Conference on Information and Knowledge Management (**CIKM**), 2021.

10. Cansu Sen, Thomas Hartvigsen, Biao Yin, Xiangnan Kong, Elke Rundensteiner. *Human Attention Maps for Text Classification: Do Humans and Neural Networks Focus on the Same Words?* Annual Meeting of the Association for Computational Linguistics (**ACL**), 2020.

9. Thomas Hartvigsen, Cansu Sen, Xiangnan Kong, Elke Rundensteiner. *Learning to Selectively Update State Neurons in Recurrent Networks.* ACM International Conference on Information and Knowledge Management (**CIKM**), 2020.

8. Jidapa Thadajarassiri, Cansu Sen, Thomas Hartvigsen, Xiangnan Kong, Elke Rundensteiner. *Learning Similarity-Preserving Word Meta-Embedding*. IEEE International Conference on Big Data (**IEEE BigData**), 2020.

7. Erin Teeple, Thomas Hartvigsen, Cansu Sen, Kajal Claypool, Elke Rundensteiner. *Clinical Performance Evaluation of a Machine Learning System for Predicting Hospital-Acquired Clostridium Difficile Infection.* International Conference on Health Informatics (**HEALTHINF**), 2020.

6. Cansu Sen, Thomas Hartvigsen, Xiangnan Kong, Elke Rundensteiner. *Patient-Level Classification of Clinical Note Sequences Guided by Attributed Hierarchical Attention.* IEEE International Conference on Big Data (**IEEE BigData**), 2019.

5. Cansu Sen, Thomas Hartvigsen, Xiangnan Kong, Elke Rundensteiner. *Learning Temporal Relevance in Longitudinal Medical Notes.* IEEE International Conference on Big Data (**IEEE BigData**), 2019.

4. Jidapa Thadajarassiri, Cansu Sen, Thomas Hartvigsen, Xiangnan Kong, Elke Rundensteiner. *Comparing General and Locally-Learned Word Embeddings for Clinical Text Mining.* IEEE International Conference on Biomedical and Health Informatics (**BHI**), 2019.

3. Thomas Hartvigsen, Cansu Sen, Elke Rundensteiner. *Detecting MRSA Infections by Fusing Structured and Unstructured Electronic Health Record Data.* Journal of Communications in Computer and Information Science (**CCIS**) 1024, 2018.

2. Thomas Hartvigsen, Cansu Sen, Sarah Brownell, Erin Teeple, Xiangnan Kong, Elke Rundensteiner. *Early Prediction of MRSA Infections using Electronic Health Records.* International Conference on Health Informatics (**HEALTHINF**), 2018.

1. Cansu Sen, Thomas Hartvigsen, Kajal Claypool, Elke Rundensteiner. *CREST - Risk Prediction for Clostridium Difficile Infection Using Multimodal Data Mining.* European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (**ECML**), 2017.

Contents

1 Introduction

1.1 Motivation

Time series data are found in a wide variety of domains including Electronic Health Records [98], Human Activity Recognition [93, 120], Cyber-security [112], Weather prediction [97], Remote Sensing [80], Astronomy [29], Seismology [33], Finance [65], Criminal Investigation [99], and Neuroscience [12]. For example, as a patient stays in a hospital, her physiological data are collected over time, forming complex high-dimensional time series data which are useful to many medical applications [39].

Due to the prevalence of time series data, the broad field of Time Series Data Mining [30] studies methods for extracting insights from time series. Time series data collection has skyrocketed over the last decade [79] from a wide variety of new domains. Data from new domains often come with their own unique properties and challenges and have spurred the study of new problems. These problems broadly include, but are not limited to, Time Series Classification [5], Forecasting [1], Change-Point Detection [2], Indexing/Segmentation [106], Anomaly Detection [16], and Clustering [71].

Time series classification (TSC) in particular is one of the most challenging time series mining tasks [28, 132]. In TSC, we seek to predict the class labels of previously-unseen time series instances. Applications of TSC include *in-hospital acquired infection prediction* [123], *seizure prediction from EEG data* [3], and *human activity recognition* [118], among many. The task of time series classification is to train a model that can accurately assign previously-unseen time series to a set of known classes. In the standard setting, values of each time series are assumed to be collected at fixed intervals, where there are even gaps between observations. For example, autonomous vehicles are equipped with a

variety of sensors, each of which records data at a fixed schedule (e.g., 10 measurements per second). Naturally, a time series is represented by a sequence of timestamp, value pairs (t, v), though when the gaps between observations are regular, the timestamps are often ignored. On the contrary, observations are often *irregular*: timestamps may not increase with equal steps and can no longer be ignored. This more-general case is extremely common and we refer to this property as *Observation Timing*. For example, in a hospital a clinician requests different measurements at different times while they work to understand a patient's health. This results in highly irregular time series that are also multivariate. There may be different numbers of observations made between two different time series or variables. Together, these features present a major challenge to modern machine learning methods, which expect fixed-length input features.

Further, most of the time, all time series in a given dataset are assumed to be fully-observed: each full time series is assigned one label and is classified based on the entire series. For example, when classifying which patients in a hospital acquired COVID during their stay last year, there is no notion of *when* in a patient's time series to generate a prediction.

However, in online time-sensitive settings, we require models that can model time series while they are being collected. For example, when classifying a patient's risk of acquiring COVID *while* they are in the hospital, classifications must be made based on only *some* of their ongoing data, with a preference for earlier predictions. We refer to this property of a model as its *Prediction Timing*.

As illustrated in Figure 1.1, many challenges of real-world time series classification stem from two types of timing, which we identify and study in this dissertation:

1. **Prediction Timing.** How quickly a classifier comes to a decision directly impacts how useful it is in time-sensitive domains. For instance, a classification that is made too late does not allow a user enough time to react. However, determining when

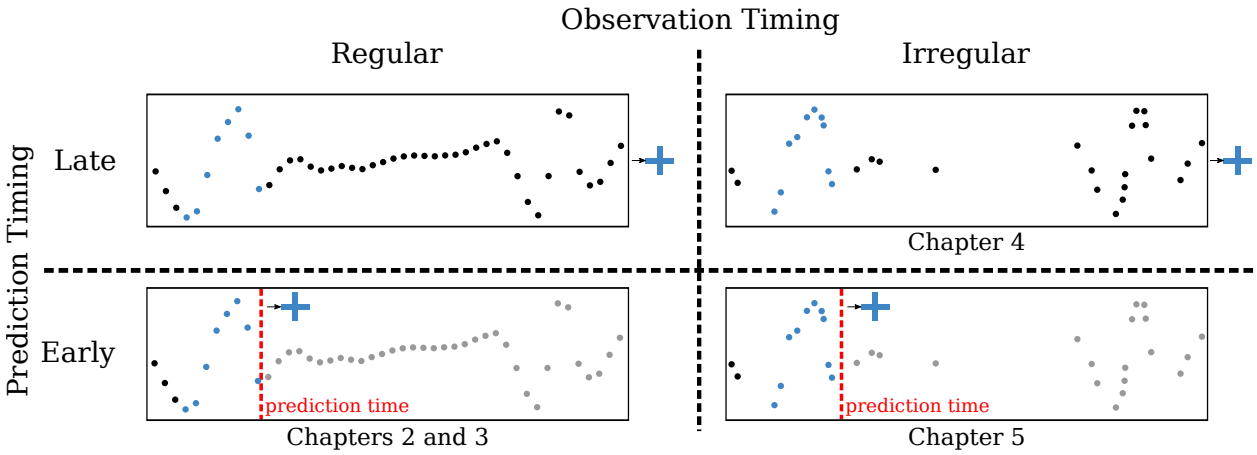


Figure 1.1: Directions of study in this dissertation. Vertical red dashed lines indicate *when* the classification is made and + indicates prediction of the positive class. Each rectangle contains a time series, in which time progresses from left to right Most prior time series classification (TSC) is made only after observing all timesteps and assumes regularly-spaced observations. Chapter 5 tackles the ultimate integrated goal, but solving it requires solutions to problems in Chapters 2, 3, and 4.

a classifier *should* return its prediction is largely unstudied. In fact, it is currently left to practitioners to protect users from such problems on a case-by-case basis. As more machine learning solutions are sought in time-sensitive domains, a more attractive solution is adaptive classification where the classifier can *learn* the importance (urgency) of the early timing of a prediction, here called *prediction timing*.

2. **Observation Timing.** Time series observations are frequently recorded at irregular intervals. Modeling the relationship between the data-collection process and the final classification task is a challenging task. An ideal TSC method can not only effectively classify time series with irregular observations but *leverage* patterns in the *observation timing*.

1.2 Prediction Timing: Early Classification of Time Series

The usefulness of a time series classifier is often driven by the time at which a prediction is generated. This insight is made exceedingly clear by the vast array of work on Early Classification of Time Series (ECTS) [126, 128, 127, 72, 50, 35, 88, 87, 46, 47, 82, 105, 26, 130]. ECTS is the problem of finding *early* timesteps at which a classifier can stop processing and return its prediction, ignoring all future observations. This is essential in time-sensitive domains, where *actionable* classifications are those that are made early enough to allow a user to react to the prediction. For example, in clinical diagnosis if a model's prediction of an infection is made only after a patient's symptoms are exceedingly obvious, the model is not adding value to the clinical decision making process. In this example it does not matter how accurate the predictions are because they are not useful with respect to *prediction timing*. All solutions to this problem involve deciding at each timestep whether or not to stop and generate a classification. To-date, they all assume timesteps to be evenly-spaced. This ignores the relationship between the size of gaps between observations and the cost of waiting for more data. Intuitively, if a model decides to wait for the next observation, but the observation arrives far into the future, this can violate the goal of making predictions as early as possible.

1.2.1 State-of-the-Art Early Classification of Time Series

The importance of ECTS has driven the development of research in early classifiers. The main directions of study are *Shapelets* [128, 127, 72, 50, 35, 130], *Classifier Ensembles* [88, 87], and *Deep Reinforcement Learning* [46, 47, 82].

Early work on ECTS began through *shapelet* discovery, where discriminative subsequence exemplars are found within a time series dataset. Finding these shapelets typi-

cally involves exhaustive search through all possible subsequences. This does not scale to large numbers of time series, long time series, or high dimensional time series. For example, [128] first extracts all possible subsequences from the given time series dataset, then uses distance-based classification to find out which subsequences are most discriminative. At test time, if a subsequence within a new time series instance matches a predefined shapelet well-enough (according to a hand-picked distance metric), the time series is associated with the respective class and the classification is returned early. Such an approach is neither scalable nor *tunable* as there is no clear way to balance between the goals of earliness and accuracy. Instead, they simply preserve accuracy.

The most recent work has focused instead on *prefix-based* ECTS [88, 87, 82, 95, 25, 78], seeking *discriminative time series prefixes*. This scales to multiple variables more readily. Some works use Classifier Ensembles combined with pre-defined Stopping Rules such prediction confidence [88, 87, 25]. The most recent approaches train Deep Learning models that solve multi-objective optimization problems during training [46, 47, 82, 95]. These prefix-based approaches are scalable to multivariate time series and allow, for the first time, *tunability* between the contradictory goals of *earliness* and *accuracy*.

1.2.2 Challenges of Early Classification of Time Series

Early classification of time series is a challenging problem for many reasons. First, to be useful in practice, a solution to this problem must be *tunable* between the respective emphasis on *earliness* versus on the *accuracy* of the predictions. These objectives typically contradict one another and a balance between them must be struck. The challenges of this problem underlie the deeply-studied field of *multi-objective optimization* [81, 24, 64]. This trade-off exists in every application of an ECTS algorithm, and therefore tuning the goals of an early classification of time series method should be made sufficiently intuitive for

the users of such a system that may have domain insights about these requirements.

Second, the relationship between *prediction time* and *usefulness* is often not explicitly known. Instead, the only feedback available for training an early classification of time series method is whether or not the classification task was solved correctly. Since learning when to make a prediction is not directly supervised, a solution to early classification of time series must learn to integrate goals of accuracy and earliness together.

Third, discriminative information may be distributed both over time and over multiple variables in the multi-variate case. A good solution must thus be able to combined information from possibly vastly-different times across variables to recognize multi-variate signals.

1.3 Observation Timing: Irregular Time Series

Most time series classifiers assume that observations are made at regular intervals [30]. However, this assumption of regular *observation timing* is unrealistic in many domains. A clear example of this is in monitoring physiological streams of patients in a hospital where observations may be both irregular in *when they are recorded* and in *which variables are recorded at a given time* [10, 77]. By learning from these data directly, we can avoid transforming the irregular time series to regularly-sampled time series (often through binning), thus reducing estimation error in regions without observations and summarization error in regions with many observations [69]. As such, classifying such irregular time series has also recently gained attention by the machine learning and data mining communities [10, 6, 32, 38, 107].

The standard approach to classifying irregular time series is to forcibly transform the observations into regularly-sampled time series prior to applying methods that assume regularly-sampled inputs, such as in [74]. This transformation is often achieved through

binning: A set of fixed-sized and evenly-spaced bins are hand picked, then bins with multiple observations are summarized into one value and empty bins have values imputed [15, 138]. This throws away much of the valuable information in irregular time series and injects bias directly into the classification process by mandating hand-picked sampling rates. For instance, high sampling rates incur high estimation error because most bins have no observations. Low sampling rates result in heavily-summarized time series, deleting fine-grained signals. Finding this balance is more art than science and performance decays on both ends of the spectrum.

1.3.1 State-of-the-art Irregular Time Series Classification

Due to the rising prevalence of irregular time series data, there have been many recent works studying irregular time series classification [32, 38, 107, 69, 10]. The vast majority of irregular time series classifiers rely on extensive preprocessing where the observation timings are first forced to be made regular before the modeling can be applied. This is typically achieved by hand picking a preferred "reference" time steps (which to-date are all regularly-spaced), then estimating values at each reference timestep. This estimation has been done using Gaussian Process models [32, 38, 107, 69], linear interpolation, continuous-time RNNs [10, 15], and neural ordinary differential equations [101]. Some recent works have also avoided picking reference steps manually by only considering timesteps where at least one variable is observed and estimating all other variables [116, 6]. This approach incurs high estimation error as many values need to be predicted. Additionally, some works have begun investigating the use of Differentiable Set Learning [135] to classify Irregular Time Series [54], although to-date they only serve to accelerate training time while unfortunately decaying classification performance.

However, in order to classify irregular time series, each of these methods still relies

on hand picking a set of reference timesteps at which to estimate new values. To-date reference timesteps are always chosen to be evenly-spaced over time so that the final classifier can assume fixed-length inputs. By definition, if few reference timesteps are chosen, global trends may be captured (such as slope), while local trends (such as short-term signals) may be ignored. On the contrary, if many reference timesteps are chosen, local trends are captured, but the new time series representations are at risk of high estimation bias in the presence of noise. On both sides of this spectrum exist time series representations that are detrimental to the classification task. It is currently left up to the user to avoid the pitfalls associated with both ends of the spectrum and there is no widely accepted and principled approach to picking a sampling rate beyond trial and error using validation data.

1.3.2 Challenges of Irregular Time Series Classification

Much of time series classification assumes regularly-sampled time series due to the challenges associated with the irregular time series setting: (1) Irregular time series are by nature challenging to represent using fixed-length vectors, even though many powerful techniques rely on such representations. (2) Time series can be noisy and task-relevant regions are often confined to small regions. This challenge is compounded by the fact that how noisy the time series are also depends on how the data are represented (fine-grained sampling techniques can make the task-relevant regions harder to find in the presence of noise). (3) ITS are not naturally represented as fixed-length feature vectors, as are required as input data format by most time series classifiers. There are no clear rules in implementing efficient batch-processing methods for such data. (4) Modeling *when* events occur is a challenging continuous-time prediction problem, as observed by the vast amount of work on *temporal point processes* [22]. Additionally, in time series classification, modeling *when*

events occur can often be useful information itself [75, 15]. However, there is no general consensus on precisely how to incorporate this information into the classifier in the irregular time series case. Some recent methods have incorporated intensity functions [107], though they do not include ablation studies on the effects of their modeling choices.

1.4 Overview of tasks of this dissertation

This document has two parts: The first half studies Early Classification of Time Series for *regular* time series. The second half focuses on classifying *irregular* time series.

Chapter 2: Tunable Early Classification of Time Series. We seek a method for *tunable multi-class* ECTS, allowing a user to choose how much to emphasize between earliness and accuracy. Following previous approaches, we also tackle the multi-class ECTS problem, which is to find one timestep per time series at which a classification is made. This assumes there is only one label associated with each time series. We also assume regularly-spaced observations, as is done in all previous ECTS work [126]. This *tunable* setting is a key advance beyond most state-of-the-art ECTS algorithms since the best balance between these contradictory goals must be task-specific. In this chapter, we develop the first tunable approach to ECTS [46].

Chapter 3: Tunable Early Multi-Label Classification of Time Series. Many real-world classification problems can be best described in terms of multi-label predictions, as opposed to the multi-class case. In the multi-label setting, we assign a *set* of labels to a given time series instance. As in ECTS, to make these predictions *actionable* they should be made as *early* as possible. We once again assume regularly-spaced inputs. The key challenge and opportunity of this problem is in leveraging the correlations between the labels themselves to make earlier and more accurate label-set predictions. For this prob-

lem, we first define the open problem of Early Multi-label Classification and then develop its first solution: a model that predicts label sets while using as few timesteps per class as possible [47].

Chapter 4: Continuous-Time Attention for Irregular Time Series Classification. Many time series are naturally recorded at *irregular* intervals. This is especially true in clinical applications involving physiological streams where clinicians monitor patient health [107, 70]. In this context, I propose the study of *continuous-time attention*, where a model should explicitly learn *moments of interest* that are most relevant to classification. Such a model should be robust to noisy inputs and less sensitive to estimation error associated with classic binning approaches to irregular time series analysis. While targeted at applications with small discriminative regions, a good solution should still be able to handle long-term discriminative trends, such as slopes by observing enough information along the entire timeline. In this dissertation, we propose the first method for classifying long and irregular time series via an attention mechanism [49].

Chapter 5: Early Classification of Irregular Time Series. Once we can find relevant signals for different classes in continuous time we can train a model to classify ongoing irregular time series *early*. Despite this natural line of questioning, this new setting adds many new challenges of its own, particularly because previous we combine early classification of time series methods require regularly-spaced inputs—including our own methods introduced in Chapters 2 and 3. In this problem, "take one step forward in time" has an entirely new meaning because there may be a large gap in time before the next timestep.

An effective solution to this problem therefore requires us to model *ongoing* time series with irregular gaps between observations, a state-of-the-art and notoriously-challenging

problem for existing machine learning methods. In this chapter, we combine early classification of time series with learning from irregular time series to define a new family of problems [44].

1.5 General Approaches

In general, we develop deep reinforcement learning methods for time series classification problems of both prediction and observation timing. For prediction timing, we train reinforcement learning agents that decide whether or not to stop and classify ongoing time series as data are collected sequentially. We use recurrent neural networks to produce vector representations of ongoing time series. For observation timing, we develop continuous-time recurrent networks that can create vector representations of ongoing series, even in gaps between observations. Further, we introduce reinforcement learning agents that search for small and relevant portions of irregular time series, even when they are small.

1.6 Contributions

This dissertation makes several contributions to machine learning and data mining research. 1) We are the first to formalize the complementary notions of observation and prediction timing, which together cover substantial group in time series classification research, opening doors to extensions of current methods into new directions. 2) We kick off the study of reinforcement learning for problems of prediction timing, an approach that has gained traction with the community and is now the state-of-the-art approach to making early and accurate classifications for time series data. 3) We generalize early classification to the multi-label setting, introducing new notions of prediction timing and

opening doors to develop models that are more practical. 4) We intersect prediction and observation timing to introduce a new family of time series classification models that is much closer to the real needs of time-sensitive decision makers.

1.7 Organization

PART 1: PREDICTION TIMING

- *Chapter 2.* We develop the first tunable approach to early classification of time series, which was also the first of many deep learning methods for early classification of time series [46].
- *Chapter 3.* We define the open problem of early multi-label classification and propose its first solution [47].

PART 2: PREDICTION TIMING

- *Chapter 4.* We propose the first method for classifying irregular time series where only a small portion of the timeline is relevant to the classification task [49].
- *Chapter 5.* We combine early classification of time series with learning from irregular time series to define a new family of problems: early classification of irregular time series [44].

CONCLUSIONS AND FUTURE WORK

- *Chapter 6.* We draw conclusions across the work presented in this dissertation, summarizing the contributions of our work.
- *Chapter 7.* We introduce promising directions that will follow the research discussed in this document.

2 Tunable Early Classification of Time Series

2.1 Introduction

2.1.1 Background

As introduced in Chapter 1, traditional time series classification assumes that a time series as a whole has been received before predicting its class label [30]. In time-sensitive applications, however, it is essential that predictions are generated well before the entire series has been observed. In these settings, decision-makers must determine how much accuracy to sacrifice in favor of earliness, with the optimal trade-off depending on both the task and the domain.

2.1.2 Motivating Example

Figure 2.1 depicts an example of the Early Classification of Time Series (ECTS) problem where each time series contains unique signals indicating their respective class labels. Approach 1 illustrates the traditional classification scheme, predicting labels only after the entire time series has been observed. This results in a highly accurate classifier, as it has the opportunity to capture all signals at the cost of providing predictions at the *very end* (indicated by the dashed halting-line). Approach 2 refers to the strict early classification method, choosing a *fixed early timestep* at which to always stop and predict. In this approach, for some time series signals have not arrived yet, while for others decisions are postponed unnecessarily. Approach 3 shows the benefit of *adaptive* early classification, selecting halting-points on a case-by-case basis (vertical line) thus allowing for early, yet accurate, predictions.

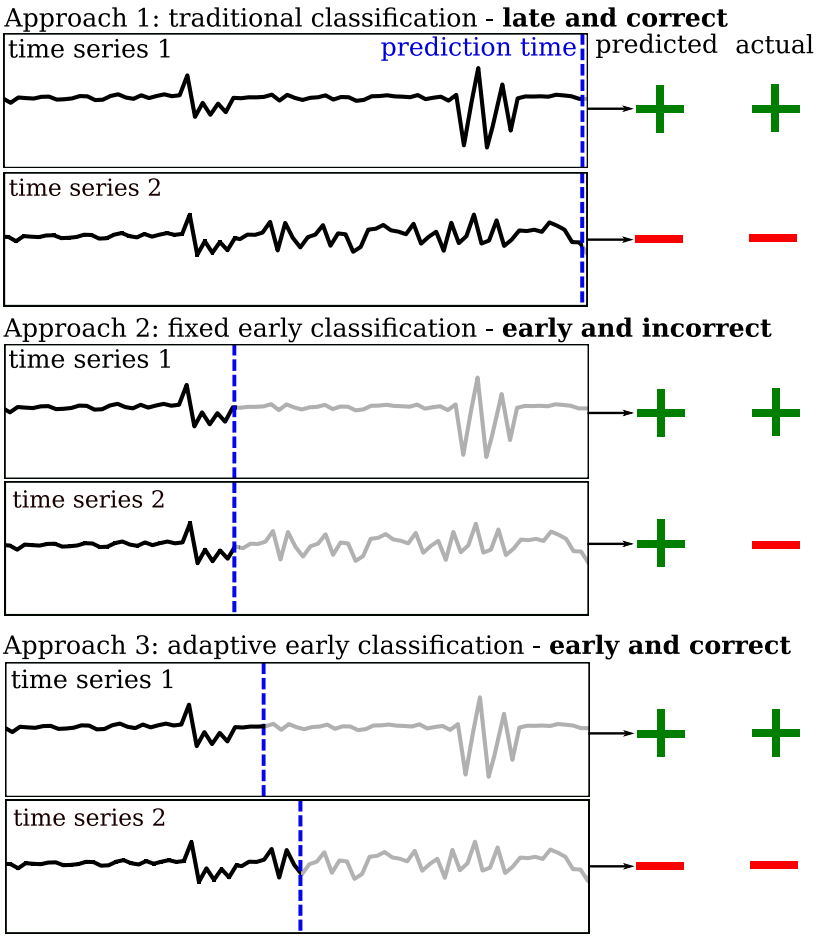


Figure 2.1: Example of three approaches to early classification of two time series. + and – denote class labels; vertical dashed lines indicate halting-points. Timesteps after halting-points in gray are not used for classification.

We note that an effective model for time series classification in time-sensitive domains should not only model discriminating signals, but also identify timesteps at which enough information has been observed to reliably (to the requested degree) predict a label. It must also be tunable based on the desired domain-specific emphasis on accuracy versus earliness.

2.1.3 Problem Definition

As introduced in Chapter 1, Section 2.1, the ECTS problem is to select a timestep in a time series at which enough information has been observed to predict a class label. It is challenging to balance the number of observed timesteps with the expected accuracy. This is because such multi-objective optimization problems involve task-dependent trade-offs. A good solution for one time series may be bad for another, requiring solutions to be highly adaptive and data-driven.

2.1.4 Challenges

Despite the importance of ECTS, several open challenges remain. We expand on the major challenges introduced in Chapter 1 as follows:

- *Multiple conflicting objectives*: Earliness and accuracy tend to contradict one-another. A maximally-early classifier may not have enough information to make accurate predictions, while a late classification may cause unnecessary delay and miss precious opportunity to react. The balance is task-specific and an optimal trade-off depends on the particular task and domain. So far, no method allows for direct tuning between these two goals.

- *Lack of supervision*: There are no labels indicating where signals occur within a time series; instead the complete time series is typically labeled by its class. Thus quantifying whether or not a prediction *should* be made at a particular timestep is difficult. In short, ECTS contains an inherently unsupervised sub-task within an otherwise supervised learning problem.

- *Multivariate signal evolution*: In multivariate time series, signals indicative of a particular class label may develop at vastly different times across variables, making the

identification of one halting point per time series (composed of all variables) harder. There has been little work [35, 50, 72] in the multivariate setting of ECTS, each with limited scalability.

2.1.5 Proposed Method: EARLIEST

In this work, we propose a solution to the aforementioned open challenges called **E**arly and **A**daptive **R**ecurrent **L**abel **EST**imator, or short, EARLIEST. EARLIEST is a novel deep network composed of a recurrent neural network (RNN)-based *Discriminator* with a reinforcement learning-based stochastic *Controller* network. During classification, the recurrent model generates representations of time series one timestep at a time, capturing complex temporal dependencies. The controller interprets these in sequence, learning to parameterize a distribution from which decisions are sampled at each timestep, choosing whether to *stop and predict a label* or *wait and request more data*. Once the controller decides to halt, the discriminator interprets the sequential representation to classify the time series. By rewarding the controller based on the success of the discriminator and tuning the penalization of the controller for late predictions, the controller learns a *halting policy* which controls online halting-point selection. This results in a learned balance between *earliness* and *accuracy* depending on how much the controller is penalized. The size of the penalty is a parameter chosen by a decision-maker according to requirements of the task.

2.1.6 Contributions

In contrast to traditional sequence-matching ECTS methods, our model-based approach supports flexible earliness-accuracy trade-offs per task using one integrated parameter, being optimized for earliness and accuracy together in one end-to-end model. The resultant solution corresponds to a general deep network model applicable to a rich variety

of time-sensitive classification tasks, including video [78, 122] and text [56]. Empirical studies on real-world tasks demonstrate that our approach outperforms baseline methods while providing effective balancing between opposing goals.

2.2 Related Work

As introduced in Chapter 1, this Chapter proposes the first work supporting task-dependent tunability in early classification of time series through joint-optimization, supporting both univariate and multivariate data. This work relates to early classification of time series methods—see Chapter 1—and conditional computation in neural networks.

Conditional Computing. Conditional computation in neural networks deals with learning when to activate different subsets of neural networks, depending on input data [103]. This can reduce the extensive computation required to train a neural network since fewer computations need to be made per example [11]. Additionally, the depth of a neural network has a major impact on performance [20], but selecting the proper network complexity remains empirical and is often more art than science. Our model leverages the idea of selectively activating parts of a neural network and can be viewed as longitudinal conditional computation: learning when to activate sections *in time*. There is one other halting RNN, built for text classification [56], which uses multiple loss functions without the full reinforcement learning setting. [21] uses reinforcement learning for ECTS but does not model temporal dynamics of the time series.

Table 2.1: Basic Notation

Notation	Description
N	Number of time series in dataset.
M	Variables per time series.
L	Classes for prediction.
T	Number of possible timesteps.
$X_t^{(i)}$	Variables at timestep t for time series i.
$y^{(i)}$	True label for time series i.
$a_t^{(i)}$	Action at timestep t for time series i.
$p_t^{(i)}$	Prob. of halting at timestep t for time series i.
$S_t^{(i)}$	Learned representation for $X_{0,\cdots,t}^{(i)}$.
$\pi_\theta(\cdot)$	Policy, maps states to actions: $\pi_\theta(S_t) = a_t$.
$\tau^{(i)}$	Chosen halting-point for time series i.
	where $i = 1, \cdots, N$ and $t = 1, \cdots, T$.

2.3 Methods

2.3.1 Problem Definition

Given a set of labeled multivariate time series, $\mathcal{D} = \{(X, y)\}$ containing N time series instances and labels, consider the i^{th} instance

$$X^{(i)} = \begin{bmatrix} | & | & & | \\ x_1^{(i)} & x_2^{(i)} & \cdots & x_T^{(i)} \\ | & | & & | \end{bmatrix}$$

where $x_t^{(i)} \in \mathbb{R}^M$ contains the M variables recorded at time t. Henceforth, for ease-of-reading, we describe our method for one time series and omit index i when it is not ambiguous. The aim is to learn parameters θ of a function $f(\cdot)$, which maps a time series X to a label $\hat{y}$ (*i.e.* $f_\theta(X) \rightarrow \hat{y}$), ultimately generalizing to classify time series not observed during training. During the training process, the goal is to match predicted labels $\hat{y}$ to

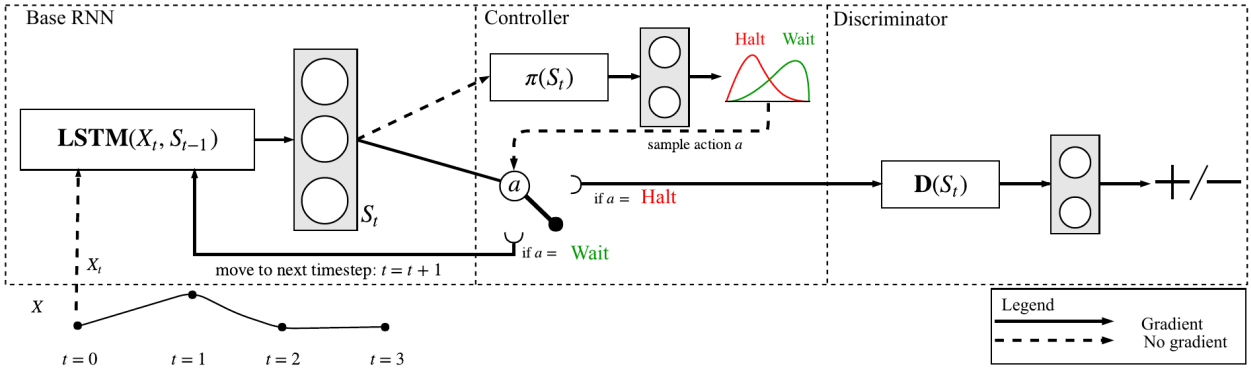


Figure 2.2: Overview of EARLIEST. Selected action a chooses whether or not to pass S_t to the *Discriminator* or back to the *Base RNN* to process the next timestep. Dashed lines indicate no gradient flow through these paths.

their corresponding true labels y where $y \in Y$ denotes the label associated with X and $Y = \{0, \cdots, L\}$, the set of possible class labels. Each time series is associated with exactly one label.

As an example, for in-hospital adverse-event detection, a multivariate time series $X^{(i)}$ may contain a patient's vital signs recorded longitudinally throughout her stay. This instance is labeled positive, or $y^{(i)} = 1$, if an adverse event occurs. Otherwise, $X^{(i)}$ belongs to the control group and $y^{(i)} = 0$.

In this work, we model f_θ as a combination of neural networks. However, as opposed to using all T timesteps to generate this prediction, for each time series we seek a timestep $\tau \leq T$ that is both small enough to satisfy a preset requirement for earliness and large enough to satisfy a requirement for successful classification. We refer to the selected τ as the *halting-point*.

2.3.2 RNN & LSTM Background

RNNs have emerged as the state-of-the-art for many time series analysis models [42, 84] and other sequence modeling tasks such as sequence generation [129, 41]. Our proposed model builds on RNNs, which construct vector representations for real-valued sequences.

At each step of a sequence, a new representation is learned via a function of the previous representation and new data observed at the current step. The final vector, computed at the final step and modeling dynamics of the sequence, can then be used to classify the sequence. Empirically, RNNs with Long Short-Term Memory (LSTM) cells [52] are more effective than as they were originally proposed [27] as they preserve information over longer sequences.

2.3.3 Proposed Method

The aims of our proposed adaptively-halting RNN, named EARLIEST, are twofold. First, to model multivariate time series for classification, and second, to select a *halting-point* at which enough timesteps have been observed to make a task-dependently adequate prediction. EARLIEST is a deep neural network consisting of three sub-networks: (1) a *Base RNN* which learns to model multivariate time series, generating low-dimensional vector represenations, (2) a *Discriminator Network*, or *Discriminator*, which learns to predict class labels based on the *Base RNN*'s model, and (3) a *Controller Network*, or *Controller*, which decides at each step whether or not to halt the *Base RNN* and activate the *Discriminator*. Once the *Controller* chooses to halt, the processing of the current time series is complete. An overview of the EARLIEST architecture is shown in Figure 2.2, where we display a rolled-up version of the RNN, showing the interaction between each network for each timestep.

The *Discriminator* is trained with respect to the classification task while the *Controller* is rewarded based on the success of the *Discriminator* and is punished based on how many steps it takes before deciding to halt. Thus, the *Controller* and *Discriminator* learn to cooperate to make correct predictions. To incorporate earliness, we add to the final objective function a loss term that competes with the *Controller*'s natural tendency to wait,

thus balancing the trade-off between accuracy and earliness according to the scale of this loss term. The final output of EARLIEST is a label $\hat{y}$ which is generated at some *halting point* τ, where $\tau \leq T$. The tunability of the model dictates how much less τ is than T, which affects the accuracy of the model depending on where signals are located.

Base Recurrent Neural Network

An RNN augmented with LSTM cells [52] rests at the heart of EARLIEST, mapping variables observed at each timestep, X_t, to vector representations $S_t \in \mathbb{R}^k$ where k is the number of hidden dimensions, a tunable hyperparameter. Standard to RNN literature, we refer to the whole recurrent part of the network simply as an LSTM. One vector S_t is created per timestep and is referred to as the *hidden state*. Each vector S_t summarizes the time series dynamics present in $X_{\{0,\cdots,t\}}$. Since these vectors inform the other parts of the network, we refer to this recurrent component as the *Base RNN*.

The LSTM is a function which learns to represent time series data as vectors. Hidden state vector S_t is computed as a function of currently-observed data X_t and the previous hidden state S_{t-1}, hence the recurrent nature of the model. In an LSTM, the computation of S_t relies upon the computation of a cell memory state C_t, which is then used to compute hidden state S_t. The LSTM's success comes from learned gating mechanisms that curate information contained in vector C_t. To compute C_t, first a *forget* gate controls what information to remove from previous cell state C_{t-1}:

$$f_t = \sigma(W_f \cdot [S_{t-1}, X_t] + b_f) \tag{2.1}$$

An *input* gate controls new information added to C_t:

$$i_t = \sigma(W_i \cdot [S_{t-1}, X_t] + b_i) \tag{2.2}$$

The updated C_t is then computed as the gated combination of memory state C_{t-1} and current X_t using the *forget* and *input* gates:

$$C_t = f_t \odot C_{t-1} + i_t \odot \eta(W_c \cdot [S_{t-1}, X_t] + b_c) \tag{2.3}$$

Finally, state representation S_t is computed through an *output* gate shown in Equation 2.4 operating on a non-linear C_t in Equation 2.5.

$$o_t = \sigma(W_o \cdot [S_{t-1}, X_t] + b_o) \tag{2.4}$$

$$S_t = o_t \odot \eta(C_t) \tag{2.5}$$

S_t is then used to inform decisions made by the *Controller*, generate classifications by the *Discriminator*, and compute the next hidden states S_{t+1} if the *Controller* so chooses. In these equations, W's and b's are learnable parameters, $\eta(\cdot)$ is the hyperbolic tangent function, σ is the sigmoid function, $\cdot$ is the dot product, and $\odot$ represents the hadamard product. For conciseness, we group these parameters into one large matrix θ_b. We denote this entire process as function LSTM$(\cdot)$ such that LSTM$_{\theta_b}(X_t, S_{t-1}) = S_t$. While we use LSTM cells, it is also possible to use alternative gating-mechanisms such as the Gated Recurrent Unit [19].

Controller Network

The *Controller* is a reinforcement learning agent that decides whether or not to halt the *Base RNN* at each timestep, prompting the prediction of a label. To achieve this goal, the *Controller* solves a Partially-Observable Markov Decision Process (POMDP) where at each timestep observations arrive from a state, an action is sampled using a learned policy, and a reward is observed according to the selected action's quality. The objective

is to optimize long-term rewards according to success of the *Discriminator*. The model is trained by gradient-based policy search.

State: At each timestep t, the state is the set of currently observed time series variables X_t, essentially a slice across all variables at timestep t. Taking advantage of the representational power of the *Base RNN*, the hidden state S_t is used as an observation from this state space. S_t informs the selection of an action by the *policy*.

Policy: Next, an action is selected by a stochastic policy $\pi_{\theta_c}(S_t) = a_t$, which treats input S_t as immutable data. In our experiments, we use a one-layer fully-connected neural network to approximate this function. Typical to reinforcement learning, we sample the action from a parameterized distribution. Thus, a learned function maps S_t to p_t, where p_t is the probability of halting, computed as

$$\begin{aligned} p_t &= \sigma(W_{ha}S_t + b_{ha}) \\ &= \frac{e^{W_{ha}S_t + b_{ha}}}{e^{W_{ha}S_t + b_{ha}} + 1} \end{aligned} \tag{2.6}$$

where W_{ha} and b_{ha} are learnable parameters for mapping hidden outputs to actions and σ is the sigmoid function, which ensures outputs between zero and one. p_t then parameterizes a Bernoulli distribution from which action a_t is sampled according to $P(a_t = 1) = p_t$. In addition to hidden state S_t, we use current timestep t as additional context to the network when computing p_t.

Actions: Sampled action a_t dictates the proceedings of the *Base RNN* as follows: if $a_t = 0$, the *Controller* has selected WAIT. This prompts the Base RNN to move forward one timestep, the action-selection process beginning again with LSTM$(X_{t+1}) = S_{t+1}$. On the other hand, if $a_t = 1$, the *Controller* has selected HALT, at which point the *Discriminator* is activated to predict a label and processing of the current time series ends. Once the controller selects HALT (or if $t = T$), t is considered to be the halting point τ. We use ε-

greedy action selection to avoid abundant exploitation in the *Controller*: with probability ε, action a_t is replaced with a random action and exponentially decrease ε from 1 to 0 during training, as shown in Equation 2.7. During training, ε exponentially decreases from 1 to 0.

$$a_t = \begin{cases} a_t, & \text{with probability } 1 - \epsilon \\ \text{random action}, & \text{with probability } \epsilon \end{cases} \tag{2.7}$$

Reward: To train the *Controller*, it must observe returns which qualify the parameters of the current policy. To encourage cooperation between the *Controller* and *Discriminator*, this return takes the form of a reward that quantifies the success of the *Discriminator*. Thus, when the *Discriminator* is correct, we set reward $r_t = 1$, and when it is incorrect, r_t = -1. The objective of the *Controller* is to maximize total reward $R = \sum_{t=0}^{\tau} r_t$.

Discriminator Network

The *Discriminator* generates a prediction $\hat{y}$ by first projecting the hidden state S_t into L-dimensional space using a fully-connected layer. Next, the resulting vector is normalized to sum to one via the softmax function and can be treated as probabilities. This computation is shown in Equation 3.11 where W_{ho} and b_{ho} are parameters for mapping the hidden state to the output space and are grouped into matrix θ_d.

$$\begin{aligned} \mathrm{P}(Y = i \mid S_t, W_{ho}, b_{ho}) &= \text{softmax}(W_{ho} S_t + b_{ho}) \\ &= \frac{e^{W_{ho} S_t + b_{ho}}}{\sum_j e^{W_{ho} S_t + b_{ho}}} \end{aligned} \tag{2.8}$$

Since the output vector sums to one after the softmax function, predicted label $\hat{y}$ is

simply the maximum probability:

$$\hat{y} = \arg\max_i \mathrm{P}(Y = i \mid S_t, W_{ho}, b_{ho}) \tag{2.9}$$

Training

In the training phase, the goal is to iteratively update all learnable parameters of EARLIEST, minimizing errors made by the *Discriminator* and maximizing the rewards observed by the *Controller*. For readability, we gather all learnable parameters of EARLIEST into matrix θ. EARLIEST is optimized by minimizing one loss function $J(\theta)$, shown in Equation 2.15, using stochastic gradient descent (SGD). The *Base RNN* and *Discriminator* are optimized together with respect to cross entropy loss shown in Equation 4.3 where θ_{bd} indicates parameters of the *Base RNN* and *Discriminator*.

$$J_{bd}(\theta_{bd}) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \tag{2.10}$$

In contrast to the other networks, in training the *Controller* the goal is to find parameters θ_c that attain the highest expected return:

$$\theta_c^* = \arg\max_{\theta_c} \mathbb{E}[R] \tag{2.11}$$

Since the *Controller* involves sampling actions, back-propagation does not directly apply, mandating transformation from this raw form to a surrogate loss function [114]. This objective can thus be optimized using gradient descent by taking steps in the direction of $\mathbb{E}[R \nabla \log \pi(S_{0,\cdots,\tau}, a_{0,\ldots,\tau}, r_{0,\cdots,\tau})]$ [104]. The gradient can then be approximated as shown in Equation 4.4, which can then be minimized to update the parameters of the controller

[124].

$$J_c(\theta_c) = -\mathbb{E}\left[R\sum_{t=0}^{\tau}\log\pi(a_t|S_t)\right] \tag{2.12}$$

However, minimizing $J_c(\theta_c)$ directly leads to gradient estimates that change dramatically across examples, resulting in high variance in policy updates since each example is treated as if in isolation [114]. To handle this, we add a baseline to $J_c(\theta_c)$, similar to [85], so that θ_c is updated based on *how much better the observed reward is than average*, resulting in

$$J_c(\theta_c) = -\mathbb{E}\left[\sum_{t=0}^{\tau}\log\pi(a_t|S_t)\left[\sum_{t'=t}^{\tau}\big(R - b_{t'}\big)\right]\right] \tag{2.13}$$

where b_t is predicted at each timestep. We learn this baseline by reducing the mean squared error between b_t and R, forcing b_t to approximate the mean R. In our implementation, we use a one-layer baseline network with a ReLU nonlinearity (Equation 2.14) that observes the same information as the controller (hidden state and timestep) and outputs one real value b_t at each timestep t.

$$b_t = \max(0, W_b[S_t, t] + b_b) \tag{2.14}$$

Balancing Earliness and Accuracy

Up to this point, the *Controller*'s only objective is to maximize the performance of the *Discriminator*. To add earliness as a goal, we employ an additional loss term, shown as the last term of our final loss function $J(\theta)$ in Equation 2.15. This loss term encourages halting, depending on hyperparameter λ. When λ is large, to minimize the loss the probability of

selecting HALT must be large, controlling earliness directly.

$$J(\theta) = J_{bd}(\theta_{bd}) + \alpha J_c(\theta_c) - \lambda \sum_{t=0}^{\tau} \log \pi(a_t = 1 \mid S_t) \tag{2.15}$$

Thus, since minimizing the log probability corresponds to increasing the probability, by increasing λ, we effectively increase emphasis on HALT. On the other-hand, when λ is small or 0, it leaves the *Controller* free to exclusively maximize the performance of the *Discriminator*. We note that in some cases, this may not mean observing all timesteps. For example, if a time series is too long, the LSTM may have trouble remembering relevant information. Altogether, this loss term creates competition on the optimization of the *Controller*'s parameters as they are tugged in opposite directions, the force of the tugging being controlled by λ.

There are other ways we can trade off between earliness and accuracy. For example, this balance could be embedded directly into the reward. We find that empirically our solution is superior, though this direction deserves more study.

2.4 Experiments

2.4.1 Datasets

We evaluate our method on four variants of a synthetic dataset and three real-world time-sensitive datasets from different domains.

`SimpleSignal`: The true locations of signals within time series are rarely known. To better understand both how EARLIEST performs in classification and in halting when it sees signals, we create a synthetic dataset. Here, we can record exactly where signals are located. For, each time series is initialized with a zero at each of the T timesteps. Then, for positive examples, we sample a location $t \in \{0, \ldots, T\}$ from a selected distribution

and substitute a one at timestep t. Negative examples remain all zeros. The selection of the signal distribution aids us in studying how each method captures the true signal locations in a variety of settings. We use four signal distributions: *uniform, normal, right-skewed*, and *left-skewed.* In this setting, a successful model should halt when a signal is observed (a one) *and* generate the correct prediction. Given access to each full time series, a classifier should achieve perfect accuracy.

`Mortality`: This use case concerns predicting in-hospital mortality and is based on a real-world dataset composed of EHR records from over 58,000 intensive care unit stays in the Beth Israel Deaconess Medical Center (publicly-available MIMIC-III database [59]). These EHR records contain time series of vital signs and microbiology tests. For clinical classification tasks (*e.g.,* diagnosis), early predictions allow clinicians to take actions that directly benefit patient well-being. The task here is to predict which patients will perish during their stay given their multivariate time series of vital signs and lab tests. We extract patients with positive *Mortality Flags*, indicating their deaths and randomly draw an equal number of surviving patients from the rest of the database. This leads to health records from 11,508 patients. We use the five most frequently recorded vital signs as our variables. The fine-grained variables aren't often recorded at the same time, leading to sparse data. To handle this, we compute daily averages for each variable, fill missing values with variable-wise means, and use the first ten days for each series. Each stay consists of a multivariate time series where each variable is a vital sign, microbiology test, laboratory result, etc. Considering all possible variables that may be observed for a patient leads to an incredibly high-dimensional setting, motivating the need for feature selection. Therefore, we examine a set of commonly-recorded tests and vital signs from tables `MICROBIOLOGYEVENTS`, `CHARTEVENTS`, and `LABEVENTS`, pick the five most frequent variables in this cohort (which as expected end up all being from `CHARTEVENTS`), listed in Table 2.2 along with the `ITEMID`'s, which identify measurements for their extrac-

tion. Since the timestamps across variables are often misaligned, we take hourly averages, impute missing values with variable-wise means, and center each variable around zero before classifying the time series.

Variable	ITEMID
Systolic Arterial Blood Pressure	51
Motor Response	454
Non-invasive Blood Pressure Mean	456
Temperature (F)	678
Non-invasive Blood Pressure Alarm [Low]	5817

Table 2.2: Variables from the CHARTEVENTS table in MIMIC-III

Seizures [3]: Seizure activity detection from EEG records, this EEG data set from 500 subjects, each of whom had their brain activities recorded via EEG, is used. There are a total of 11,800 178-timestep time series, and the task is to detect which of these time series contains evidence of epileptic seizure activity. Since there are only 2,300 cases of such activity, we down-sample an equal number from the negative class, resulting in a balanced dataset with 4,600 time series. Finally, we center the time series around zero and compute the mean value of every 10-timestep chunk, summarizing each 178 timestep series into 17 final timesteps.

TwitterBuzz [60]: To predict buzz events on Twitter, we work with this data set of 77-dimensional labeled time series indicating whether or not a spike in tweets on a particular topic is observed. Starting with over 140,000 timesteps, we compute the mean of every five steps, center the time series around zero, and break the resulting 28,000 timesteps into 2,800 length-ten sequences. We then extract the 1,271 time series containing any buzz events and balance the dataset by randomly selecting an equal number of no-buzz time series, resulting in a balanced dataset of 2,542 time series.

2.4.2 Compared Methods

We compare EARLIEST's performance to the following algorithms:

- *LSTM-Fixed* [123, 78]. Fixed halting-point selection is common in time-sensitive classification tasks. It requires that an analyst pre-selects a timestep at which all classifications will be made. Since EARLIEST uses an LSTM, we use a fixed halting-point version of LSTM, referred to as *LSTM-Fixed*.

- *LSTM-s* [78]. Designed for early classification of video, *LSTM-s* can be directly applied to time series. *LSTM-s* encourages early confidence in its predictions by penalizing the model when it becomes less confident similar to an LSTM version of ECTS algorithm ECDIRE [88]. Similar to LSTM-Fixed, this method also uses a fixed pre-selected halting-point.

- *LSTM-Confidence*. Classifiers based on a softmax output assign a probability to each class [14]. For this baseline, we set a threshold α for the minimum confidence of a probabilistic prediction. Once the network surpasses this confidence (*i.e.* $\max \hat{y} > \alpha$), the model halts and predicts the most likely class. This model adaptively-halts per time series, but since α is not included in the loss function, parameters are not updated with respect to the goal of earliness.

Since other multivariate ECTS algorithms [35, 50] do not directly support multiple trade-offs, our model is not directly comparable.

2.4.3 Implementation Details

For all datasets, we use an 80% training, 10% validation, and 10% testing split. We use the training set to learn model parameters and the validation set to evaluate the performance of a particular hyperparameter setting (*e.g., nodes-per-layer* or *learning rate*). The

training and validation sets are used multiple times to tune hyperparameters, then the testing set is used once to compute the final accuracy of each model. We use a two-layer BaseRNN for these experiments, first learning 10-dimensional embeddings for time series variables, and second learning sequences of 10-dimensional representations, one per time step. We repeat this setup five times and compute averages over these five settings to compute final results. The model is optimized using Adam [63] with a learning rate of 10^{-4}. All models are implemented using PyTorch with the code available at `https://github.com/thartvigsen/EARLIEST`.

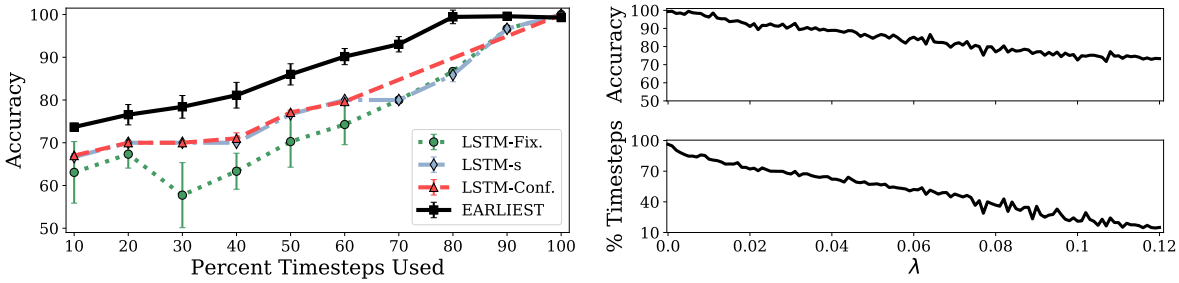
2.4.4 Experimental Results

Experiments on Synthetic Data.

We first evaluate the performance of EARLIEST in a controllable setting where signal locations are known using the synthetic dataset `SimpleSignal` described in Section 4.4.1. We evaluate EARLIEST in two ways: by determining how early and accurate EARLIEST is compared to our baselines by controlling the earliness-accuracy trade-off hyperparameter λ; and second, how effectively EARLIEST halts when it observes signals, thus matching the true distribution of signal locations.

Accuracy and timing: EARLIEST should more accurately classify instances earlier than the baseline methods due to adaptive-halting. In Figure 2.3, EARLIEST is run for $\lambda \in [0.0, 0.15]$. λ does not directly control accuracy or earliness, instead urging the optimization in one direction or the other. Thus for each λ, EARLIEST stabilizes at some accuracy and distribution of halting points. We extract the mean accuracies and halting-points (computed as the average percent of timesteps used, or $\frac{\tau}{T}$) with baseline predictions made at the same time. We see in Figure 2.3a that for nearly all halting-points, EARLIEST significantly outperforms the baselines, indicating higher accuracy using on average the same

information. The only overlap is when all models have observed the entire time series, leading to perfect classification accuracy. Additionally, we report the sensitivity of EARLIEST to parameter λ in Figure 2.3b. This shows the average accuracy and percent timesteps used for each λ. The smooth down-ward trends indicate that as λ is increased, there is a smooth coverage of all possible halting-points in these time series.



(a) Accuracy and timing

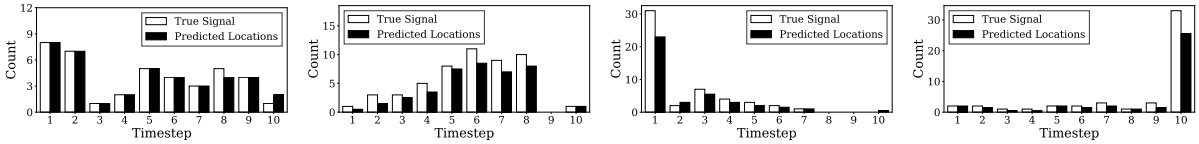
(b) λ coverage in EARLIEST

Figure 2.3: Accuracy and prediction times on synthetic data. (a) EARLIEST makes predictions more accurately and earlier than baselines. (b) λ-tuning leads to smooth halting at each timestep.

Signal-capturing: Next, EARLIEST should halt when it sees a signal, and wait otherwise. To understand if this is the case, we compute the root mean squared error (RMSE) between EARLIEST's selected halting points and the true distribution of signals, thus quantifying how well EARLIEST halts when it sees a signal.

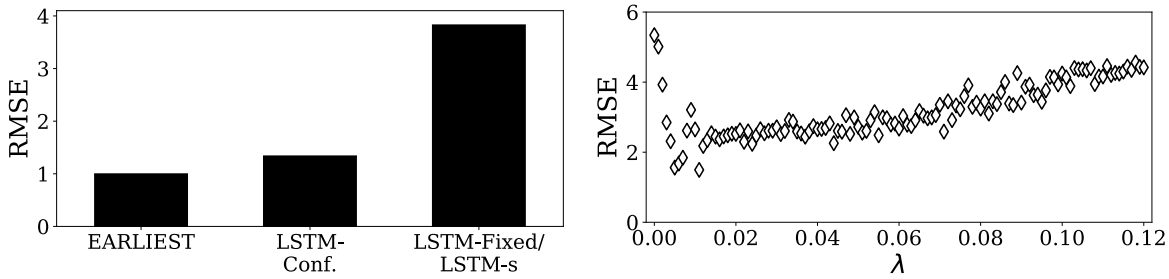
We use four distributions of positive labels to generate four versions of the `SimpleSignal` data set. We expect that EARLIEST should halt when it observes a positive signal and otherwise wait until the end of the time series to classify negative instances. We deem the final timestep to be the true signal location for negative examples so we compare positive predicted locations to their known signal locations, measuring true-positive signal-capture.

We show EARLIEST's signal-capturing on `SimpleSignal` with *uniform*, *normal*, *left-skewed*, and *right-skewed* signal distributions in Figure 2.4, where λ is fixed to be 0.014, the



(a) Uniform (b) Normal (c) Left-skewed (d) Right-skewed

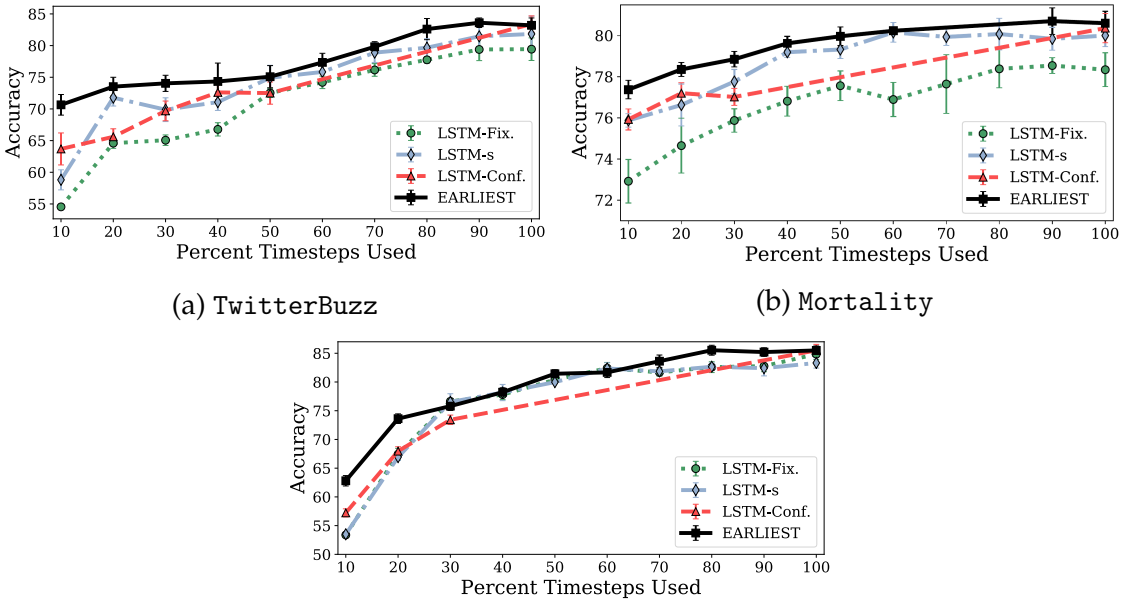
Figure 2.4: *True Signals* indicates where signals actually appear in the time series, *Predicted Locations* shows the true-positive halting-points selected by EARLIEST.



(a) Baseline comparison. (b) EARLIEST signal-capture sensitivity to λ.

Figure 2.5: Signal-capturing capabilities on synthetic data. RMSE compares the predicted locations of positive examples with the true signal locations. (a) Minimum RMSE between predicted and actual locations for each model. (b) Parameter analysis for EARLIEST.

best performing λ on the *uniform* signal distribution. Additionally, this setting empirically tended to result in a wide variance in predicted locations. We show the true-positive predictions for each distribution, and the halting-points are averages over all experiment repetitions. In the *Uniform* setting, signals are equally likely to appear at any timestep. In Figure 2.4a we see that the bars match, indicating that EARLIEST does capture signal locations despite having no access to this information. We see a similar trend for the *Normal* signal distribution ($\mu = 6.0$, $\sigma = 2.0$) in Figure 2.4b, though the signal capture is not as exact. The *left-skewed* distribution tests whether or not EARLIEST halts when observing consistently-early signals. In Figure 2.4c we see this is the tendency of the model, though EARLIEST waits until the end to make one prediction once, missing the signal location. Using the *right-skewed* distribution we test whether or not EARLIEST can wait for long periods of time if it does not observe any signals. In Figure 2.4d we show



(a) `TwitterBuzz` (b) `Mortality`

(c) `Seizures`

Figure 2.6: EARLIEST's performance on real-world data. EARLIEST consistently has equal or better accuracy than the compared methods given on average the same information. Error bars are standard deviation over five repetitions.

this is in fact the case, and the distributions match quite well. These results demonstrate that EARLIEST is capable of flexibly capturing signal locations, halting when signals are observed.

We next compare EARLIEST's signal capture to that of the baselines using the uniform distribution of signal locations. For a fair comparison, we compare the best average performance of each method. In Figure 2.5a we show that EARLIEST with $\lambda = 0.014$ dramatically outperforms *LSTM-FH* and *LSTM-s* and is slightly superior to *LSTM-Confidence*, demonstrating that EARLIEST is better at halting when it observes signals. We show the effect of parameter λ on RMSE in Figure 2.5b. As expected, RMSE is poor with both low λ (emphasizing waiting) and high λ (emphasizing halting), and better in between. This indicates that beyond controlling accuracy, λ also controls how effectively EARLIEST halts and captures signals.

Experiments on Real-world Data.

We next present results using real-world datasets `Morality`, `Seizures`, and `TwitterBuzz`. We compare accuracies and average locations in Figure 2.6. Each point for EARLIEST represents averaged results from λ settings that lead to average halting at each timestep. Overall, we see that EARLIEST consistently performs equal to or better than the compared algorithms at all possible halting points. Optimal halting-points are unknown for these datasets, so we compare accuracy and earliness.

For `TwitterBuzz`, we observe in Figure 2.6a that EARLIEST outperforms the baselines at nearly all timesteps. In the $20 - 60\%$ range, LSTM-s performs equally well. EARLIEST shows an average increase of 2.81% accuracy over the best among the baselines at each timestep with a maximum of up to 12.88%. This indicates that some timesteps significantly benefit from adaptive-halting.

For `Mortality`, we see in Figure 2.6b that EARLIEST consistently has a higher accuracy than the compared methods. This results in a more modest average increase of $.96\%$ over the best baselines with a maximum improvement of 1.91%. Interestingly, despite fine-grained search, no λ led to EARLIEST halting in the $70 - 80\%$ range, possibly due to underlying time series dynamics.

Finally, for `Seizures`, in Figure 2.6c we again see a similar trend, the largest difference being that the most improvement over the compared methods occurs early in the time series. This may indicate that signals in this dataset appear early, and after a certain point each model has observed nearly all useful information. In these experiments, EARLIEST shows a mean of 2.61% improvement over the best baselines with a max of 9.77%.

From our experiments on real-world datasets, we conclude that for many parameter settings EARLIEST has higher accuracy than the baselines while using on average fewer timesteps. For all settings, EARLIEST performs equally or better than the baselines.

LSTM-s is competitive in many settings, though this method suffers from its requirement for a preset fixed halting-point. For LSTM-Confidence, the resulting halting-points are erratic since the confidence-threshold is set externally to the model. We conclude that benefits of adaptive-halting are also strongly-dependent on the timing of signals. We suspect that the most benefit may be seen with uniformly-distributed and pronounced signals.

2.5 Conclusions

In this work, we have developed EARLIEST, an adaptive model for the early classification of time series. Our neural network-based approach tackles the unsupervised nature of early classification through reinforcement learning. EARLIEST directly models the multiple objectives of early classification, accuracy and earliness, allowing for their joint optimization despite conflicting tendencies. During classification, our model learns representations of multivariate time series that are then used to both inform early-halting decisions and to predict labels. Our experimental results for both synthetic and real-world datasets demonstrate that EARLIEST effectively learns to halt when it observes a signal and wait otherwise, leading to fine-tuned reactive case-by-case signal-capture. EARLIEST effectively balances earliness and accuracy via one hyperparameter, allowing for analyst-controlled task-dependent solutions.

3 Tunable Early Multi-label Classification of Time Series

3.1 Introduction

3.1.1 Background

Early Classification is the crucial task of predicting class labels of time series as early as possible for time-sensitive applications such as healthcare [45] and transportation [43]. In many cases, predictions made late are simply useless, regardless of their accuracy. Meanwhile, many time-sensitive tasks can be best modeled as multi-label classification, given that multiple classes (*e.g.,* diseases) may be assigned to one instance (*e.g.,* patient). However, standard multi-label classification methods rely on observing an entire time series prior to its classification [121]. We refer to the intersection of time-sensitive and multi-label learning as the Early Multi-label Classification problem, where a successful model must accurately predict the correct set of labels for a time series while observing as few of its values over time as possible.

An important example of Early Multi-label Classification is diagnosing a patient's infections since the very sick often acquire multiple infections concurrently. Evidence of said infections may appear at different times throughout a patient's stay in an intensive care unit, and the likelihood of developing one infection often depends on which infections a patient has already acquired. A successful diagnosis model should thus capture the dependencies between observed infections *while* predicting each infection *as soon as enough evidence has been observed*. The earlier a correct diagnosis is predicted, the more

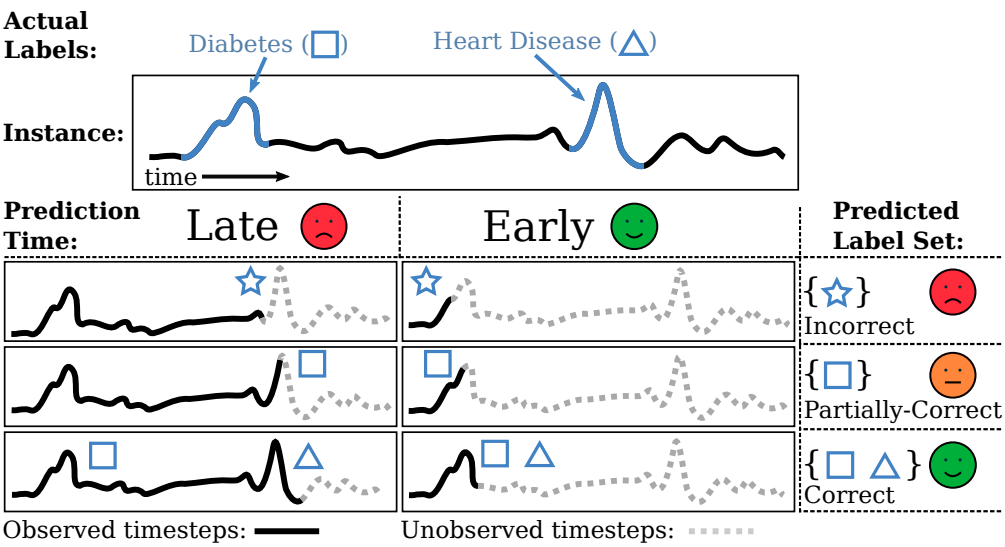


Figure 3.1: Early Multi-label Classification Problem. Multiple labels can be assigned to each instance of time series data. The correct labels must be predicted as early as possible. In this example, the bottom right box is ideal: The correct label set is predicted after observing very few timesteps.

time clinicians have to react and intervene, thus improving patient outcomes. A concrete example of this setting is depicted in Figure 3.1 where the optimal outcome is achieved through the early and accurate prediction of both Diabetes and Heart Disease.

3.1.2 State-of-the-Art

Recently, major progress has been made in tackling the early and multi-label problems independently.

Early Classification.. As introduced in Chapter 1, *Early Classification* has gained significant attention, particularly for applications using time series data [46, 128, 127, 72, 50, 35, 87], though initial work has also been done on text [56] and video [78]. Most recently, *tunable* Early Classification [87, 82, 46] has garnered much interest since the balance between *earliness* and *accuracy* tends to be task-dependent. However, these proposed methods have only studied *multi-class* classification (one label per instance), ignoring the crucial relationships between the labels that are inherent to such multi-label classification

settings.

Multi-label Classification. Multi-label Classification has also recently seen a surge of interest, in particular the study of *Classifier Chains* [17, 121, 133, 117, 91, 131, 90]. This approach aims to directly model the conditional probability between predicted labels, often using Recurrent Neural Networks (RNN). A key limitation of these works is that they predict label sets only *after* observing the entire instance, directly contradicting the requirements of Early Classification. Another restriction of popular approaches [121, 91, 131] is that they require that a pre-defined label order be provided a-priori to enable optimization [90, 117, 17]. While this simplifies the problem, it unfortunately limits application to domains with easily-defined label orderings, such as speaker diarization [31]. In the context of most time series datasets, it is rarely known precisely at which timestep the evidence of a label arises. Finally, the integration of earliness into the complex multi-label context remains unexplored.

3.1.3 Problem Definition

In this work, we are the first to address the open problem of Early Multi-label Classification, which is to predict the correct label set of a time series instance while observing as few timesteps per class as possible. This results in one early timestep per class, at which point its prediction is made. While the evidence for a time series' class labels may appear at any time step, the 'true' timestep of each class label is entirely unsupervised – the only supervision comes from one label set for the entire time series. The halting timesteps should be *dynamic*, varying depending on the time series. The crux of the problem is that making predictions early is essential for each class, but there may not be enough early evidence to warrant a high-confidence prediction, thus defining a multi-objective optimization problem. An effective solution must leverage relationships between labels to

predict accurate label sets, even in the absence of clear class signals.

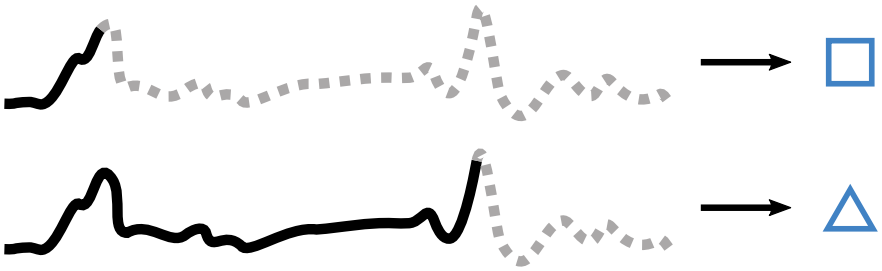
3.1.4 Challenges

Despite the importance and potential impact of Early Multi-label Classificaiton, open challenges remain:

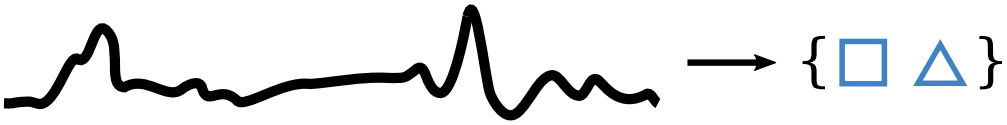
- *Unknown label timing*: For multi-label classification, a label set is available for each time series indicating its associated classes (*e.g.,* recording which infections a patient acquired). However, rarely are the *steps at which class labels appear* recorded in conjunction with a time series. Thus, we may have no a-priori knowledge of when a class *should* be detected. Learning *when* to predict each class is an *unsupervised sub-problem* within an otherwise-supervised learning task.

- *Conflicting objectives*: Early classifications are typically made at the expense of prediction accuracy. Maximally-early classifications are often based on partial information, which may not be sufficient for accurate prediction. A late classification will be better-informed and thus more accurate. However, late predictions cause critical delays and thus miss precious opportunities to react rapidly. The optimal trade-off in this multi-objective problem is domain and task-driven driven.

- *Multi-label learning*: Learning the relationships between labels themselves is a challenging problem. Multi-label learning on time series while they are observed remains largely unexplored, particularly in the context of early classification where accurate label set assignment is not the only objective.



(a) Binary Decomposition [13] + Single-label Early Classification [46]



(b) Full Sequence + Multi-label Time-Series Classification [136]

(c) Early Multi-label Classification (this work)

Figure 3.2: Comparing the different solutions to the early multi-label classification problem.

3.1.5 Proposed Method: Recurrent Halting Chain

We propose a solution to the new Early Multi-label Classification problem, which we refer to as the **R**ecurrent **H**alting **C**hain, or RHC. RHC is the composition of three novel neural networks, each solving one piece of the Early Multi-label Classification problem. First, a recurrent neural network (RNN)-based *Transition Model* learns to jointly represent multivariate time series data and the conditional dependencies between the labels. This encodes multi-label learning into the classification task, acting as a classifier chain. Second, a *Disciminator* network uses the hidden state of the Transition Model to predict soft class probabilities at every timestep. Third and finally, a *Halting Policy Network* uses the soft class probabilities and the hidden state of the Transition Model to predict at each step which, if any, classes to add to the predicted label set. Once the Halting Policy Network has decided to halt all classes, no further timesteps are observed. Importantly, as soon as

a prediction is made in the time series, it is returned as an *early* prediction.

Since the true label locations are unknown, reinforcement learning allows for *dynamic* label predictions, strictly conditioned on the input data. RHC is optimized for both conflicting objectives concurrently; along the way we introduce one simple hyperparameter that trades off the emphasis in each goal. Figure 3.2 illustrates the key difference between our proposed solution and the state-of-the-art approaches to both Early and Multi-label Classification in isolation.

3.1.6 Contributions

Our main contributions are summarized below:

- We define the new open problem of Early Multi-label Classification with its roots in both Early Classification and Multi-label Classification.
- We design the first solution to Early Multi-label Classification, which advances beyond both recent deep reinforcement learning approaches to early classification and classifier chains for multi-label learning, resulting in a unified approach to this complex problem.
- Our model is evaluated on real-world time-sensitive multi-label classification tasks using several publicly-available datasets. Results show that RHC consistently beats alternate solutions in both accuracy and earliness of label prediction on a variety of settings and metrics.

3.2 Related Work

As best we can tell, ours is the first work to study the problem of Early Multi-label Classification. This direction is related to both Early Classification and Multi-label Classification.

Early Classification. The goal of Early Classification is to correctly predict the label of a time series before it is fully observed, selecting one timestep per time series at which the whole series is classified. This task is often targeted at time series data [50, 128, 127, 126, 35, 37, 36, 87], however the most recent approaches [46, 82, 25] propose a general formulation of this problem through the use of neural networks. By using neural networks, these approaches naturally model multivariate inputs [46, 82] in contrast to previous works which solely study univariate inputs [88, 127, 126, 128, 35]. The univariate approaches typically involve exhaustive search for discriminative subsequences, which scales poorly into the multivariate setting [50]. Additionally, many recent works also take a *prefix-based* approach to early classification[46, 87, 88], learning at which timestep enough information has been observed to warrant classification. This is in contrast to *shapelets* [126], which typically require exhaustive search. The prefix-based solution of "picking a halting point" can naturally be framed as a Markov Decision Process: at each timestep, decide whether or not to stop and predict the label of a time series. This observation has allowed for intuitive balancing between *earliness* and *accuracy* through reinforcement learning [46, 82]. [46] uses an RNN to model the transition dynamics of time series in conjunction with a policy network that decides at each timestep whether or not to halt the RNN and generate a prediction. [82] proposes a Deep Q Network [86] that, given a time series prefix, samples which class to predict or to simply wait for more observations. This integrates the halting and classification but does not scale as the number of classes increases since large action spaces often require too vast a number of samples [110].

A major limitation of all current Early Classification methods is that they are restricted to the multi-class setting – predicting exactly one label per time series. As shown by the wealth of multi-label learning literature, dependencies between labels in multi-label tasks can provide crucial information for solving many problems.

Multi-label Classification. Multi-label classification methods predict the labels of time series where multiple labels are possible per series. Typically, the key challenge and opportunity is in relating the labels to each other in the feature space of a learned model, a feature missed by standard multi-class algorithms. One basic approach to achieving multi-label learning is through decomposition of the multi-label problem into a set of binary classification tasks, referred to as Binary Decomposition [13]. This outputs label sets but ignores the correlation between labels and the likelihoods of different label combinations. In contrast, *Classifier Chains* have recently become a popular and intuitive approach to multi-label learning since they naturally model conditional dependency between class predictions [17, 121, 133, 117, 91, 131, 90]. This is typically achieved using an RNN that outputs labels one step at a time with its own already-predicted labels being fed back into the model at each step. A key challenge of RNN-based classifier chains is the natural requirement of a label-order with which to train the model [117, 17]. Meanwhile, the chosen label order dramatically impacts the performance of the classifier chain [119, 91]. Recent works have just begun to remove this assumption, proposing classifier chains based on confidence-ranked labeling [17] and multi-task learning [117].

The key drawback of these algorithms in time-sensitive applications is that all classes are predicted only *after* an entire sequence is observed. To achieve *actionable* decision making in time sensitive domains, predictions must instead be made at early timesteps, as observed by the Early Classification problem.

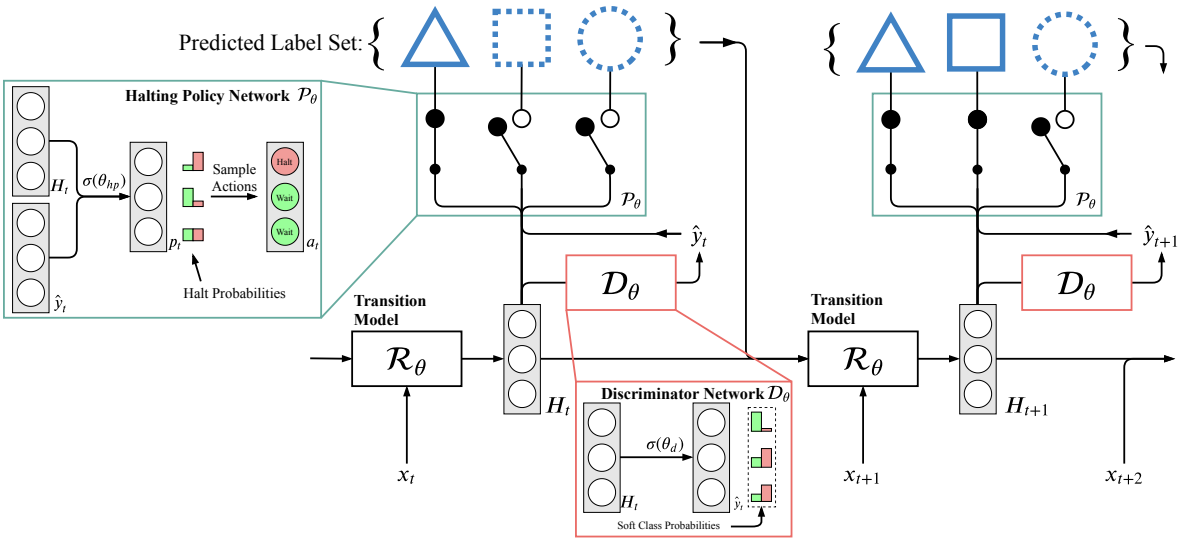


Figure 3.3: Overview of RHC. At each timestep t, the Transition Model $\mathcal{R}$ computes $H_t = \mathcal{R}(H_{t-1}, x_t, \bar{y}_{t-1})$, the new hidden state for timestep t. Using H_t, the Discriminator $\mathcal{D}$ computes soft class probabilities for every class: $\hat{y}_t = \mathcal{D}(H_t)$. Then, the Halting Policy Network $\mathcal{P}$ selects whether or not to "lock in" the predictions of each class in $\hat{y}_t$ independently in the form of one sampled action vector: $a_t = \mathcal{P}(H_t, \hat{y}_t)$. Once a class has been halted, its prediction is returned as an *early* classification.

3.3 Methods

3.3.1 Problem Definition

Given a set of labeled time series containing N time series, consider the instance $X = [x_1, x_2, \ldots, x_T]$ where $x_t \in \mathbb{R}^M$ is the M variables recorded at step t. Let $Y = [y^1, y^2, \ldots, y^L]$ denote the label set such that $Y \in \{0, 1\}^L$ is the set of L possible labels where $y^l = 1$ indicates assignment to class l. For ease-of-reading, we describe our method in terms of one time series. The learning objective is a function $f_\theta(\cdot)$ whose parameters θ accurately map $f_\theta(X) \rightarrow Y$ for series not observed during training.

As an example of this setup, consider a patient's health records collected throughout her stay at a hospital (*e.g.*, heart rate, blood pressure). While in the hospital, she is diagnosed with *diabetes* and *heart disease* but not *runner's knee*. This label set would be

Table 3.1: Basic Notation

Notation	Description
N	Number of time series in dataset.
M	Variables per time series.
L	Number of possible classes.
T	Number of steps per time series.
x_t	Values recorded at step t.
Y	True label set for one time series (*e.g.,* $\{1, 1, 0\}$).
$\bar{y}_t$	Indicator of which classes have been predicted prior to step t.
$\hat{y}_t$	Soft confidence predictions at step t.
p_t^l	Halting probability at step t for class l.
a_t^l	Halting action vector at step t for class l.
H_t	Vector representation for $X^{(i)}_{0,\cdots,t}$ from the RNN.
$\pi(\cdot)$	Policy that maps hidden states to actions.
τ^l	Predicted halting-step for class l.
	where $i = [1, \ldots, N]$, $t = [1, \ldots, T]$, and $l = [1, \ldots, L]$.

represented as, $y = [1, 1, 0]$, respectively, indicating the first two possible diagnoses were observed while the third was not. The key multi-label component is in the relationship between the labels: *diabetes* and *heart disease* often occur concurrently while *runner's knee* is independent of the other two.

The final component is in contrast to the standard multi-label classification problem: for each time series we seek one *halting step* $\tau \leq T$ per class L at which each class's prediction should be made. τ^l, the halting step for class l, must be small enough to achieve early prediction yet large enough to assign the correct label set to the time series. This requirement defines our multi-objective optimization problem since earlier predictions ($\tau \ll T$) often come at the expense of predicting correct label sets.

3.3.2 Proposed Method

We propose a Recurrent Neural Network (RNN)-based Early Multi-label Classification model. Our method, the **R**ecurrent **H**alting **C**hain (RHC), has two concurrent goals: First, to model complex time series data for multi-label classification, thereby modeling conditional dependence between labels. Second, to select one *halting timestep* τ per class at which point the model predicts the label of that class. RHC is a neural network comprised of several core components: (1) a *Transition Model* that learns to jointly represent the map of $X \rightarrow Y$ and the conditional relationship between labels *while* the labels are being predicted *in time*, acting as a *classifier chain*, (2) a *Discriminator* that predict soft confidence values $\hat{y}$ at each timestep t, and (3) a *Halting Policy Network* that decides at each step whether or not to halt each class using a *joint-learned representation* to model which classes can be predicted concurrently. Once the Halting Policy Network decides to halt a class, the Discriminator's prediction of that class is returned from the model and subsequently remains fixed for all time steps up until the Halting Policy Network has halted all classes.

The *Transition Model* and *Discriminator* are trained together as an *order-free* Classifier Chain [117, 17] since there are no labels indicating at which timesteps a class label should be predicted. The *Halting Policy Network* makes discrete decisions at each timestep (whether or not to halt and predict a class), which is non-differentiable and is trained using Reinforcement Learning, being rewarded based on how accurately the *Discriminator* predicts each class and punished according to how many steps it takes to make accurate predictions.

Transition Model

The core of RHC is a *Transition Model* $\mathcal{R}(\cdot)$, which learns joint vector representations for the time series dynamics and the conditional dependence between labels. We follow the state-of-the-art in a wide variety of sequence modeling problems and implement this component as Recurrent Neural Network (RNN) $\mathcal{R}(\cdot)$, processing input sequences one step at a time. To avoid the vanishing gradient problem pervasive in RNNs, we use Long Short-Term Memory (LSTM) [52] cells as our transition function, mapping inputs x_t to a representation H_t as follows:

$$f_t = \sigma(W_f \cdot [H_{t-1}, x_t] + b_f) \tag{3.1}$$

$$i_t = \sigma(W_i \cdot [H_{t-1}, x_t] + b_i) \tag{3.2}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \phi(W_c \cdot [H_{t-1}, x_t] + b_c) \tag{3.3}$$

$$o_t = \sigma(W_o \cdot [H_{t-1}, x_t] + b_o) \tag{3.4}$$

$$H_t = o_t \odot \phi(C_t) \tag{3.5}$$

where [ ] indicates concatenation, σ indicates the sigmoid function, $\cdot$ is matrix multiplication, and ϕ indicates the hyperbolic tangent function. W_f, W_i, W_c, and W_o represent the matrices of trainable weights for the forget, input, memory cell, and output gates, respectively. Due to the concatenation of H_{t-1} and X_t, each of these weight matrices is of shape $v \times (v + M)$ where v is the dimension of the hidden state of the RNN. Each gate is simply an affine transformation of the combination of *newly-observed information* X_t and *previous state* H_{t-1} followed by a non-linearity and so the transition function is a learned dynamical system modeling the transition of hidden state vector H.

In order to encode multi-label learning into this transition function, we use an auxiliary *indicator* vector $\bar{y}_t \in \{0, 1\}^L$ which records *at timestep t which classes have already been*

predicted, similar to [17]. Thus, $\bar{y}_t^l = 1$ indicates that class l has already been predicted and $\bar{y}_0$ is initialized as 0s prior to observing any timesteps, indicating no classes have been predicted. The transition model is thus an augmentation of the standard LSTM update equations as follows, conditioning the hidden states on $\bar{y}_t$:

$$f_t = \sigma(W_f \cdot [H_{t-1}, x_t, \bar{y}_{t-1}] + b_f) \tag{3.6}$$

$$i_t = \sigma(W_i \cdot [H_{t-1}, x_t, \bar{y}_{t-1}] + b_i) \tag{3.7}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \phi(W_c \cdot [H_{t-1}, x_t, \bar{y}_{t-1}] + b_c) \tag{3.8}$$

$$o_t = \sigma(W_o \cdot [H_{t-1}, x_t, \bar{y}_{t-1}] + b_o) \tag{3.9}$$

$$H_t = o_t \odot \phi(C_t) \tag{3.10}$$

This increases the size of the weight matrices W according to the number of classes. Thus, the Transition Model effectively captures the dynamics of the time series while it is observed while modeling the conditional dependence between labels *with respect to each other*, as is the core idea of classifier chains. Our approach thus improves upon other classifier chains in this setting by merging the time series dynamics with label correlations in the latent space of the Transition Model, effectively conditioning the model's representation on both input time series and the history of predicted labels.

Discriminator Network

The computed hidden representation H_t is subsequently projected into a probabilistic classification space through a Discriminator neural network $\mathcal{D}_\theta(\cdot)$, as shown in Equation 3.11 where $W_{ho} \in \mathbb{R}^{L \times V}$, predicting one probability for each of the possible L classes using the *sigmoid* function. Thus $\mathrm{P}(Y|H_t) \in [0,1]^L$. Importantly, H_t has been computed

with respect to $\bar{y}_{t-1}$, capturing label dependence during classification.

$$\begin{aligned} \hat{y}_t = \mathrm{P}(Y \mid H_t) &= \mathcal{D}_\theta(H_t) \\ &= \frac{1}{1 + e^{W_{ho}H_t + b_{ho}}} \end{aligned} \tag{3.11}$$

In principle, the Discriminator $\mathcal{D}_\theta(\cdot)$ can be as simple or as complicated as desired according to the complexity of the task.

Subsequently, the soft class probabilities $\hat{y}_t$ and H_t itself are sent to the *Halting Policy Network*, which predicts which of the predicted class probabilities should be halted at timestep t.

Halting Policy Network

At each step, the *Halting Policy Network* $\mathcal{P}(\cdot)$ interprets the hidden state $H_t = \mathcal{R}(X_t, H_{t-1}, \bar{y}_{t-1})$ and discretely selects which classes should be predicted at timestep t. Because there are no ground-truth halting locations, we frame this task as a partially-observable Markov Decision Process (POMDP), similar to [46, 82], which is typically solved using Reinforcement Learning. In this setup, at each step t the *state* consists of the Hidden State from the Transition Model (which represents our data and labels predicted up until step t), the possible *actions* are *Wait* or *Halt* with one action per class, and we define the rewards to be the success of classification for each class.

The first step of the *Halting Policy Network* at step t is to project the hidden representation H_t into a probabilistic space through a neural network, as shown in Equation 3.12 where $\sigma()$ is the sigmoid function and W_{hp} is of shape $L \times (v + M)$, mapping the $(v + M)$-dimensional concatenation of the hidden state and the predicted class confidences to one

halting-probability p_t per class label.

$$p_t = \sigma(W_{hp}[H_t, \hat{y}_t] + b_{hp}) \tag{3.12}$$

Importantly, this network models the joint probability of halting the prediction for each class, allowing for specific combinations of classes to be halted together, thus modeling multi-label learning in the halting component of RHC.

The predicted vector $p_t \in [0, 1]^L$ parameterizes L bernoulli distributions, one per class, from which *halting decisions* a_t are sampled. Finally, $a_t \in 0, 1^L$ where $a_t = 1$ indicates *Halt* and $a_t = 0$ indicates *Wait*, determines which classes to halt at step t. Importantly, a_t does not indicate whether or not to predict a class *positively*. Instead, $\hat{y}_t$ determines the class prediction at timestep t, which may be positive or negative. For example, if $a_t^l = 1$, indicating *halt Class l at timestep t*, the resulting prediction for the class l for time series X is $\hat{y}_t^l$, regardless of future outputs $\hat{y}_{t'}^l$ where $t < t' \leq T$.

Once a_t has been computed, $\bar{y}_t$, the vector indicating which classes have been predicted, can be updated:

$$\bar{y}_t = \bar{y}_{t-1} + a_t \odot (1 - \bar{y}_{t-1}) \tag{3.13}$$

where $\odot$ indicates the hadamard product, adding to the set of already-predicted classes maintained by vector $\bar{y}$. Thus once all classes are halted, we are left with one vector $\hat{y}$ containing the soft probabilities collected at each halting point τ^l.

The final component of the POMDP is the *reward*, which is used during training and must be designed to encourage the learned policy to achieve the desired goal. In our case, we seek a policy that leads to both *accurate* and *early* label assignments. Thus, we define the reward function as follows: for each class l, when the classification is correct,

we set reward $r_t^l = 1$, and when it is incorrect, $r_t^l = -1$. As described in Section 3.3.2, this encourages the *Halting Policy Network* to halt when the predictions will be correct and discourages halting otherwise.

A key ingredient in Reinforcement Learning is a careful balance between *exploration* and *exploitation*. To avoid policies which simply exploit actions that lead to positive rewards early on in training, we use an ϵ-greedy approach to choose between the predicted *halting decision* and a randomly-selected action, as shown in Equation 3.14 where ϵ is 1 at the beginning of training and decreases to 0 exponentially throughout training. Thus, at the beginning of training, actions are mostly random and as training proceeds, the reins are progressively handed off to the learned policy.

$$a_t = \begin{cases} a_t, & \text{with probability } 1 - \epsilon \\ \text{random action}, & \text{with probability } \epsilon \end{cases} \tag{3.14}$$

Optimizing the Recurrent Halting Chain

Our combination of supervised learning for multi-label classification with reinforcement learning for early halting requires a multi-component loss function.

The *Transition Model* $\mathcal{R}(\cdot)$ and the *Discriminator* $\mathcal{D}(\cdot)$ are jointly optimized to output class predictions $\hat{y}$ as close to y as possible by minimizing cross entropy (Equation 3.15), using standard back-propagation since all operations are differentiable, similar to [17]. To achieve this, $\hat{y}^l$ is simply the Discriminator's prediction of class l *from the timestep at which it was predicted*.

$$\mathcal{L}_{\text{sl}}(\theta) = \sum_{l=1}^{L} -(y^l \log(\hat{y}^l) + (1 - y^l) \log(1 - \hat{y}^l)) \tag{3.15}$$

This way, correct label-sets are preferred to incorrect as $\hat{y}$ is modeled as the conditional

probability between predicted labels.

Optimizing the *Halting Policy Network* is more intensive due to sampling during action-selection, though we follow the standard optimization setup for reinforcement learning agents using policy gradients. The sampling of actions in the POMDP solved by the *Halting Policy Network* is inherently non-differentiable, and so we use the standard REINFORCE algorithm [124] as a gradient estimator to train the network. The learning objective of the halting policy network is the maximization of the expected return $R = \sum_{t=0}^{\tau} r_t$:

$$\theta_{hp}^{*} = \arg\max_{\theta_{hp}} \mathbb{E}[R] \tag{3.16}$$

where θ_{hp}^{*} is the optimal parameters for the *Halting Policy Network*.

The *Halting Policy Network* samples its actions so errors cannot be propagated directly. Instead, most recent policy gradient methods transform from this raw form to a surrogate loss function [114]. The new objective can be optimized using gradient descent by taking steps in the direction of $\mathbb{E}[\nabla \log \pi(H_{0,\cdots,\tau}, a_{0,\ldots,\tau}, r_{0,\cdots,\tau})R]$ [104]. The gradient can then be approximated for the halting decisions for each class as shown in Equation 4.4. This allows for training via back-propagation but can also induce variance in the policy updates since this is not the *true* gradient of the desired objective function. To reduce said variation, we employ the standard practice of adding a baseline that approximates the expected reward to adjust the raw reward values. This way, the weights are updated with respect to how much better than average the outcomes are for each episode.

$$\mathcal{L}_{\text{rl}}^{l}(\theta) = -\mathbb{E}\left[\sum_{t=0}^{\tau^l} \log \pi(a_t^l | H_t)\left[\sum_{t'=t}^{\tau^l} \left(R^l - b_{t'}^l\right)\right]\right] \tag{3.17}$$

where b_t^l is predicted at each timestep as the output of a lightweight neural network and is forced to approximate the mean R^l via the reduction of their mean squared error.

Finally, we average the loss function in Equation 4.4 across all l classes, resulting in one final differentiable function summarizing the success of the halting policy network:

$$\mathcal{L}_{\text{rl}}(\theta) = \frac{1}{L}\sum_{l=0}^{L}\mathcal{L}_{\text{rl}}^{l}(\theta) \tag{3.18}$$

Encouraging early predictions

Finally, we enforce early predictions by minimizing the log halting probabilities according to one hyperparameter, λ, resulting in our final objective function, shown in Equation 3.19, which can be optimized using stochastic gradient descent. This extra loss term, weighted by λ, directly maximizes of the probability of halting and so as λ increases, the likelihood of halting early increases, making predictions earlier. In practice, $\lambda = 0$ is a feasible option, implying *halt only when it helps prediction*, tending towards later halting points.

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{sl}} + \mathcal{L}_{\text{rl}} + \lambda\sum_{l=0}^{L}\sum_{i=0}^{\tau^{l}}\log\pi(a_t^l = 1|H_t) \tag{3.19}$$

3.4 Experiments

3.4.1 Datasets

We evaluate our method on the following time-sensitive datasets.

`HAR` [4]: Human Activity Recognition (HAR) from smart phone data. These data consist of readings from a variety of sensors in a smartphone while 30 participants perform a set of six activities such as walking and standing. Our task is to predict *which of the activities were performed within a time window of sensor data*. Clearly, there may be multiple activities performed within one window. These data are naturally recorded with one la-

Time-Steps Observed	Evaluation Metrics	Methods				
		LSTM-BD	E-LSTM	LSTM-CC	EARLIEST	RHC [ours]
20%	Instance-AUC$^{\uparrow}$	0.88 (0.00)	0.85 (0.00)	0.90 (0.02)	**0.92** (0.00)	**0.92** (0.01)
	Micro-AUC$^{\uparrow}$	0.86 (0.00)	0.85 (0.00)	0.88 (0.01)	**0.91** (0.00)	**0.91** (0.00)
	Macro-AUC$^{\uparrow}$	0.86 (0.00)	0.84 (0.00)	0.88 (0.02)	**0.91** (0.00)	**0.91** (0.00)
	Hamming Loss$^{\downarrow}$	0.18 (0.00)	0.21 (0.00)	0.17 (0.01)	**0.13** (0.00)	**0.13** (0.01)
	Micro-F1$^{\uparrow}$	0.62 (0.00)	0.57 (0.00)	0.66 (0.03)	**0.74** (0.00)	0.72 (0.02)
	Macro-F1$^{\uparrow}$	0.62 (0.00)	0.57 (0.00)	0.65 (0.03)	**0.74** (0.00)	0.71 (0.02)
40%	Instance-AUC$^{\uparrow}$	0.91 (0.00)	0.89 (0.00)	0.92 (0.02)	**0.94** (0.00)	**0.94** (0.00)
	Micro-AUC$^{\uparrow}$	0.90 (0.00)	0.90 (0.00)	0.92 (0.02)	0.92 (0.00)	**0.93** (0.00)
	Macro-AUC$^{\uparrow}$	0.91 (0.00)	0.89 (0.00)	0.92 (0.02)	0.93 (0.00)	**0.94** (0.00)
	Hamming Loss$^{\downarrow}$	0.17 (0.00)	0.17 (0.01)	0.15 (0.01)	**0.10** (0.00)	**0.10** (0.00)
	Micro-F1$^{\uparrow}$	0.65 (0.00)	0.67 (0.02)	0.72 (0.02)	0.79 (0.00)	**0.81** (0.00)
	Macro-F1$^{\uparrow}$	0.63 (0.00)	0.68 (0.02)	0.72 (0.02)	0.79 (0.00)	**0.81** (0.00)
60%	Instance-AUC$^{\uparrow}$	0.92 (0.00)	0.93 (0.01)	0.93 (0.01)	0.94 (0.00)	**0.95** (0.00)
	Micro-AUC$^{\uparrow}$	0.92 (0.00)	0.94 (0.01)	0.93 (0.01)	0.93 (0.00)	**0.95** (0.00)
	Macro-AUC$^{\uparrow}$	0.90 (0.00)	0.94 (0.01)	0.93 (0.01)	0.94 (0.00)	**0.95** (0.00)
	Hamming Loss$^{\downarrow}$	0.13 (0.00)	0.13 (0.02)	0.13 (0.01)	0.10 (0.00)	**0.08** (0.00)
	Micro-F1$^{\uparrow}$	0.74 (0.00)	0.77 (0.03)	0.75 (0.02)	0.80 (0.01)	**0.83** (0.00)
	Macro-F1$^{\uparrow}$	0.74 (0.00)	0.78 (0.03)	0.74 (0.02)	0.80 (0.01)	**0.83** (0.01)

Table 3.2: Performance (mean (std)) of early multi-label classification on the Human Activity Recognition (HAR) dataset. "↓" indicates "the smaller the better" and "↑" indicates "the larger the better".

bel per timestep, so we split the data into 15-step time series instances and record which activities were performed within those steps. These associated activities are the instance's label set. On average each instance ends up with 25% of the possible labels. We use the Triaxial Acceleration and Triaxial Velocity from the Gyroscope in the smart phone, resulting in 490 15-step time series with 77 variables each along with 490 up-to-size-6 label sets ($N = 490$, $T = 15$, $M = 77$, $L = 6$). We set $T = 15$ to balance the number of labels per time series while maintaining a large-enough N. This does not change the distribution of labels or the locations of the signals. The distributions of class labels (Shown in Figure 3.4) are nearly balanced across all classes since this is a *scripted* dataset: during collection, each participant performed each action in sequence.

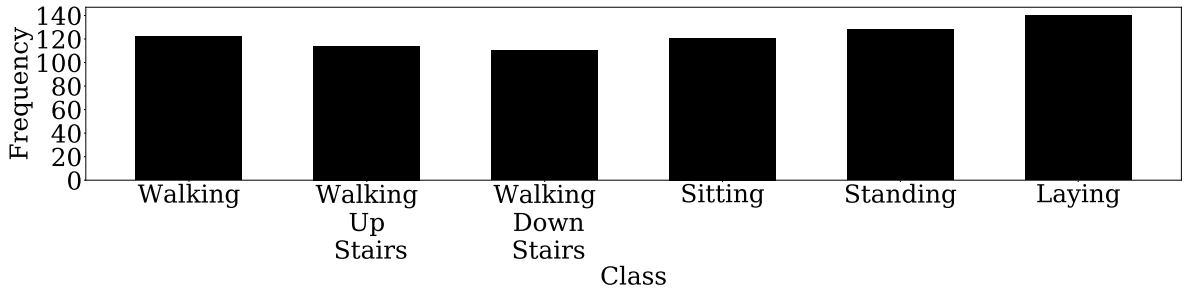


Figure 3.4: Class label balance in HAR.

ExtraSensory [118]: Similar to HAR, these data consist of smartphone sensor data recorded while 60 participants performed a variety of activities. However, since these data were collected unscripted, the label set size is much larger. Post-hoc, the labels were reduced to 52 options and each participant may have engaged in any number of these activities while carrying their smartphone. To convert these data to a multi-label time series classification task, we summarize the fine-grained sensor data by averaging the readings every ten steps and maintaining which labels occurred within those steps. This is because activities do not change much timestep-to-timestep. Then, similar to HAR, we chunk these data into ten-step sequences and record activities performed within that window using the 40 Acceleration variables. Due to label sparsity, we down-sample the 11 labels that appear in at least 1000 time series and randomly select a final set of 1000 40-dimensional time series, averaging 36% of the 11 labels per instance ($N = 1000$, $T = 10$, $M = 40$, $L = 11$). Again, $T = 10$ ensures a large enough dataset size.

Since some classes are extremely rare (for example "At the bar"), we down-sample the classes that appear in at least 1000 time series, resulting in 11 final classes: *Lying Down*, *Sitting*, *Walking*, *Running*, *Bicycling*, *Sleeping*, *Lab Work*, *In Class*, *In a Meeting*, *At Main Workplace*, *Indoors*. The frequency of these classes is shown in Figure 3.5. These frequencies are recorded from our final sample of 1000 time series. Importantly, these labels can overlap one another in interesting ways. For example, a participant could have been *Sit-*

ting while they are *In Class*, but cannot be *Lying Down* while *Bicycling*. However, since we chunk the time series into windows, it is possible that one time series is associated with both *Bicycling* and *Lab Work* if the participant rode her bike to the lab. This creates an ideal testbed for Early Multi-label Classification since some activities may be linked through concurrence (*e.g., Sitting* and *In Class*) while others may be linked causally (*e.g., Lying Down* before *Sleeping*). In the future, the use of all 52 original activities may be used to study different but related problem settings, particularly in the case of rare and highly-correlated classes.

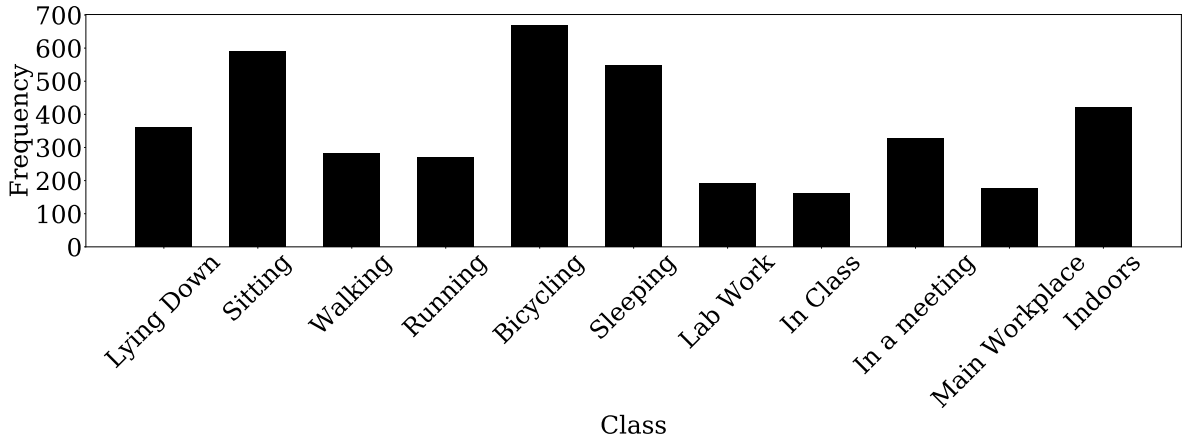


Figure 3.5: Class label balance in `ExtraSensory`.

3.4.2 Compared Methods

We compare RHC to the following algorithms, two of which are early classifiers adapted for multi-label learning, and two of which are multi-label learners adapted to early classification:

- *LSTM-BD* [52]. This method breaks the multi-label task into L binary classification tasks via Binary Decomposition [13] and achieves early classification via fixed halting-point selection [78]. Thus, *LSTM-BD* neither models label relationships nor achieves adaptive early classification.

- *E-LSTM* [25]. We augment this Early Classification method to solve the Early Multi-label Classification problem via binary decomposition. First, a threshold $\alpha \in [0, 1]$ is hand-picked prior to learning. Then, an LSTM generates a class probability $\hat{y}$ at each timestep. Once $\hat{y} > \alpha$, the classifier halts and its prediction is returned. This captures data-driven early classification (the time at which $\hat{y} > \alpha$ can vary) but this approach does not model relationships between labels.

- *EARLIEST* [46]. Our final binary decomposition baseline, *EARLIEST* uses reinforcement learning to predict a halting point at which a label prediction is made. However, this applies directly to only the *multi-class* setting. Through binary decomposition, this method outputs early label predictions but does not encode relationships between labels. Their optimization also does not capture multiple sources of reward.

- *LSTM-CC* [121]. Order-Free Classifier Chains are a recent and powerful approach to multi-label learning when true label orders are unknown (such as the Early Multi-label Classification problem). We adapt the core idea of this approach, originally designed for images, to time series. This methods first embeds a time series using an LSTM encoder. Then, an LSTM decoder predicts the labels one at a time in sequence. This method is trained to be order-free as in [17]. This approach captures the relationships between labels but requires all timesteps. To make classifications *early*, we use fixed halting points [78], forcing the model's predictions at preset timesteps.

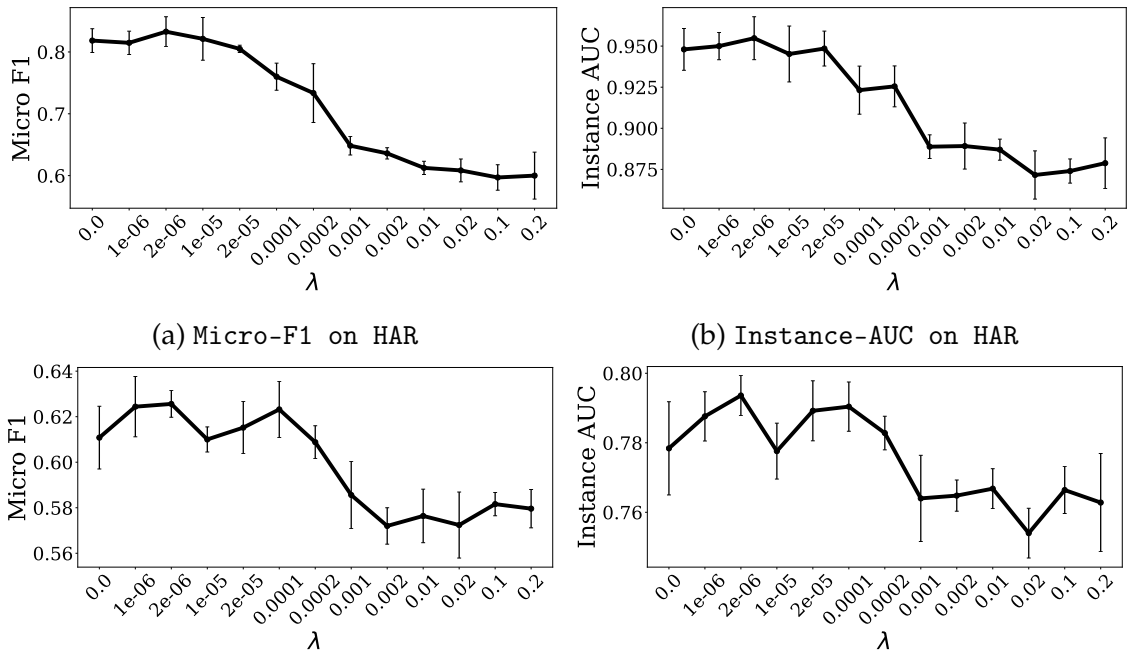
3.4.3 Implementation Details

For all datasets, we use an 80% training, 10% validation, and 10% testing split. We use the training set to learn model parameters and the validation set to evaluate the performance of a particular hyperparameter setting (*e.g., nodes-per-layer* or *learning rate*). The testing set

Time-Steps Observed	Evaluation Metrics	Methods				
		LSTM-BD	E-LSTM	LSTM-CC	EARLIEST	RHC [ours]
20%	Instance-AUC$^{\uparrow}$	0.71 (0.00)	0.71 (0.00)	0.74 (0.02)	0.71 (0.00)	**0.78** (0.00)
	Micro-AUC$^{\uparrow}$	0.70 (0.00)	0.69 (0.00)	**0.72** (0.02)	0.71 (0.00)	0.68 (0.00)
	Macro-AUC$^{\uparrow}$	0.60 (0.00)	0.62 (0.00)	0.63 (0.01)	0.63 (0.01)	**0.68** (0.00)
	Hamming Loss$^{\downarrow}$	0.31 (0.00)	0.32 (0.00)	0.31 (0.01)	0.31 (0.00)	**0.27** (0.00)
	Micro-F1$^{\uparrow}$	0.58 (0.00)	0.55 (0.00)	0.58 (0.02)	0.58 (0.00)	**0.61** (0.00)
	Macro-F1$^{\uparrow}$	0.48 (0.00)	0.47 (0.00)	0.46 (0.01)	0.47 (0.01)	0.46 (0.00)
40%	Instance-AUC$^{\uparrow}$	0.74 (0.00)	0.73 (0.00)	0.77 (0.01)	0.74 (0.00)	**0.79** (0.00)
	Micro-AUC$^{\uparrow}$	0.73 (0.00)	0.71 (0.00)	0.75 (0.01)	0.74 (0.00)	**0.78** (0.00)
	Macro-AUC$^{\uparrow}$	0.66 (0.00)	0.64 (0.00)	0.68 (0.01)	0.67 (0.00)	**0.70** (0.00)
	Hamming Loss$^{\downarrow}$	0.32 (0.00)	0.32 (0.00)	0.29 (0.01)	0.27 (0.00)	**0.26** (0.00)
	Micro-F1$^{\uparrow}$	0.56 (0.00)	0.59 (0.00)	0.60 (0.00)	0.60 (0.00)	**0.62** (0.00)
	Macro-F1$^{\uparrow}$	0.47 (0.00)	0.52 (0.00)	0.48 (0.02)	**0.53** (0.01)	0.48 (0.00)
60%	Instance-AUC$^{\uparrow}$	0.76 (0.00)	0.75 (0.01)	0.78 (0.01)	0.77 (0.01)	**0.79** (0.00)
	Micro-AUC$^{\uparrow}$	0.76 (0.00)	0.73 (0.01)	0.77 (0.01)	0.76 (0.01)	**0.78** (0.00)
	Macro-AUC$^{\uparrow}$	0.71 (0.00)	0.67 (0.01)	0.70 (0.01)	0.69 (0.01)	0.70 (0.00)
	Hamming Loss$^{\downarrow}$	0.28 (0.00)	0.32 (0.01)	0.28 (0.01)	0.28 (0.00)	**0.26** (0.00)
	Micro-F1$^{\uparrow}$	0.61 (0.00)	0.63 (0.01)	0.62 (0.01)	**0.62** (0.01)	**0.62** (0.00)
	Macro-F1$^{\uparrow}$	0.53 (0.00)	0.56 (0.01)	0.53 (0.01)	**0.55** (0.00)	0.47 (0.00)

Table 3.3: Performance (mean (std)) of early multi-label classification on the `ExtraSensory` dataset. "↓" indicates "the smaller the better" and "↑" indicates "the larger the better".

is used once to report the final evaluation metrics for each model. For all methods, we use an RNN with the LSTM transition function, learning a 20-dimensional vector representation for each time step of each multivariate time series instance. We repeat this setup five times and compute averages over these five settings to compute final results. The model is optimized using Adam [63] with a learning rate of $1e^{-2}$ and all methods are run until their loss converges, taking 200 epochs. All models are implemented using PyTorch with the code available at `https://github.com/thartvigsen/RecurrentHaltingChain`.



(a) `Micro-F1 on HAR` (b) `Instance-AUC on HAR`

(c) `Micro-F1 on ExtraSensory` (d) `Instance-AUC on ExtraSensory`

Figure 3.6: Observing the effect of λ on Micro-F1 and Instance-AUC.

3.4.4 Experimental Results

We evaluate RHC using the `HAR` and `ExtraSensory` datasets described in Section 4.4.1. We use two groups of metrics: *Instance-AUC*, *Micro-AUC*, and *Macro-AUC* to assess the ranking performance of the soft probabilistic predictions; *Hamming Loss*, *Micro-F1*, and *Macro-F1* to evaluate the hard predictions after standard rounding. Across all of these metrics, we show that RHC consistently achieves far stronger performance using fewer timesteps than the state-of-the-art alternatives described in Section 3.4.2. For both datasets, we investigate the predictions made by each method on three distinct early proportions of the time series corresponding to 20%, 40%, and 60% of the steps. In three experiments, we tune RHC and all baseline methods such that their average halting timesteps correspond to 20%, 40%, and 60% of possible timesteps.

For the `HAR` dataset, as shown in Table 3.2, RHC consistently achieves the same or better performance on all metrics at each of the three halting points. Most notably, when ob-

serving more steps (40% and 60%) RHC clearly outperforms all other approaches. This indicates that our approach effectively models the relationships between labels themselves (RHC $>$ {EARLIEST, E-LSTM, RNN-BD}) while achieving high performance with few observed timesteps (RHC $>$ LSTM-CC). In these settings, RHC achieves an average of 7.77% and 4.80% improvement, respectively, over the compared methods across all metrics. In the 20% setting, the other adaptive early halting method, EARLIEST, is quite competitive, as some metrics overlap between RHC and EARLIEST. This may suggest that at this level of partial-observability of the time series, there may not be enough evidence to relate labels to one another on these data. However, the strong *Hamming Loss* performance confirms that RHC remains superior, even when treating each task separately. Additionally, RHC's strong performance on the ranking tasks compared to the other methods indicates RHC's effectiveness in leveraging the multi-label relationships present in this dataset. Finally, we also note that LSTM-CC consistently outperforms LSTM-BD across all settings and metrics. This indicates the value of multi-label learning, even in the context of preselected halting timesteps.

We observe similar trends on the `ExtraSensory` dataset, as shown in Table 3.3. Once again, across all three early proportions of the time series, RHC consistently outperforms all other methods. In the 20% setting, RHC achieves on average 3.89% improvement over the other methods (3.32% over EARLIEST), while for 40% the improvement is 4.04%. This superiority implies that the relationships between classes themselves can be useful in achieving early classifications, shedding light on the effective pairing of the early classification and multi-label classification objectives. In the 60% setting, EARLIEST is once again competitive, resulting in a 1.1% advantage, though RHC is the best method in 4 of the 6 metrics. As demonstrated by the performance on the AUC-based ranking metrics, RHC consistently captures the multi-label relationships, appropriately ranking positive classes higher than negative classes.

Parameter Study

RHC has one hyperparameter λ that controls its emphasis on how early predictions *should* be made. We investigate its effect in Figure 3.6, demonstrating that, as expected, as λ increases, predictions are made earlier and thus the *Micro-F1* and *Instance-AUC* decrease. Importantly, λ has roughly the same effect on *Micro-F1* and *Instance-AUC*. This can been seen in Figures 3.6a and 3.6b where the trends for the `HAR` task are fairly similar to one another. The trends in Figures 3.6c and 3.6d also match each other to a significant degree. Overall, however, λ affects datasets differently, demonstrating the need for such hyperparameters.

3.5 Conclusions

In this work, we identify the new Early Multi-label Classification problem. We then design the Recurrent Halting Chain (RHC) as a solution to this problem. RHC learns to predict the label set of multivariate time series while making *early* classifications for each class, driven by reinforcement learning. RHC directly models the objectives of early and accurate label assignment jointly, achieving one integrated solution that effectively trades-off between these goals. At each timestep, RHC uses a Transition Model to represent both complex temporal dynamics in the input time series *and* conditional dependencies between labels as they are progressively predicted. The Halting Policy Network reads the hidden state at each timestep and decides whether or not each class prediction should be returned as a final classification. Across our experiments recording six metrics for three settings on two real datasets, RHC consistently outputs early and accurate multi-label classifications.

4 Attention-based Irregular Time Series Classification

4.1 Introduction

4.1.1 Background

As introduced in Chapter 1, Section 3, irregular time series (ITS) have uneven spaces between their observations and are common in impactful domains such as healthcare [73], climate science [7], and astronomy [100]. These uneven gaps arise from many sources. For example, in physiological streams, clinicians drive the collection of medical record data by requesting different lab tests and measurements in real time as they investigate the root causes of their patient's conditions [73]. *Which* measurements are taken *when* differs between patients. When classifying such time series, there are often relationships between *when* observations are made and the class label for the resulting time series. For instance, sicker patients may have more measurements.

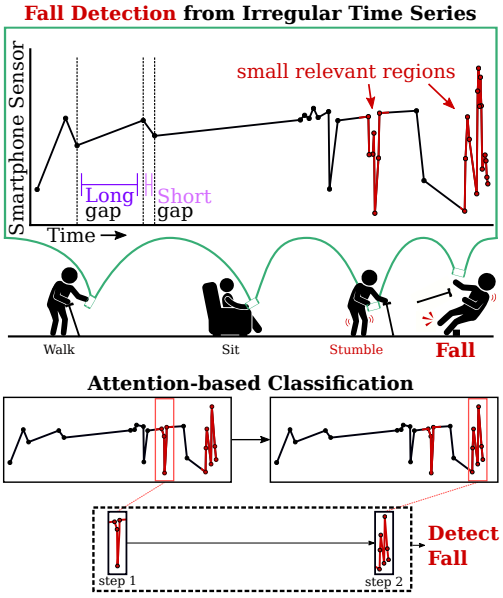


Figure 4.1: Attention-Based ITS classification.

ITS can also contain many measurements, while the regions most-relevant to the classification may take up only a small portion of the timeline, creating a small *signal-to-noise* ratio in the proportion of relevant observations.

A successful classifier must find the best moments in the continuous timeline at which to capture signals in both the values themselves and the patterns in *when* observations were made, or *informative irregularity*, while ignoring irrelevant regions.

4.1.2 Motivating Example

Consider detecting if a person *Fell* using their smartphone's sensors, as illustrated in Figure 4.1. To extend battery life, a listening probe is used to only collect data when certain conditions are met, for example when the accelerometer changes rapidly.

Since the phone is not always moving, the stored time series are naturally irregular. To detect a fall, some regions of the accelerometer's records are far more relevant than others. Leading up to a fall, for instance, a person may have stumbled earlier in the day. However, there can also be many false positives where the phone moves quickly even though the person is not falling (setting the phone down, for instance). Additionally, *when* observations are made can also be useful: if the phone moves after a long gap, the person may be getting out of bed. Since only some regions are relevant, all a classifier needs are the few most relevant moments in the timeline. Finding these moments is especially important for the long series that naturally exist in many domains.

4.1.3 State-of-the-art

There have been many recent advances in classifying ITS data, though most focus on sparse series with few observations. Many works treat ITS classification as a *missing value imputation* problem [15, 73, 138], converting ITS to regular series then performing standard classification. However, when the signal-to-noise ratio is small, many values need to be imputed to avoid aggregating intricate signals. Plus, this increases the length of the series and imputes values in irrelevant regions of the timeline. Imputing too few values

easily bypasses short signals. Some works capture informative irregularity by computing statistical features such as *missingness indicators* [73] or the time since last observations [15] as additional input variables, inflating the feature space.

Recent works have also begun learning models *directly* from raw ITS observations [107, 69, 70, 101, 23, 94]. However, they still rely on hand-picking a set of new *reference* timesteps at which to estimate values, falling prey to the same challenges of imputation. To-date, these methods do not adapt their reference timesteps to the input instances.

4.1.4 Problem Definition

In this work, we are the first to address the problem of *Attention-based ITS Classification* (ABC), which is to classify long ITS by finding small discriminative signals in the continuous timeline, as illustrated in Figure 3.1. Given a set of labeled ITS, where each series consists of one sequence of (*timestep*, *value*) pairs per variable, our aim is to produce a classifier that can correctly assign class labels y to previously-unseen instances. For any time series, the signal-to-noise ratio, or the proportion of the timeline relevant to the classification, may be very small. This is particularly true when a time series contains a large number of measurements, many of which may be made off the relevant window. A successful model should explicitly find the *discriminative moments* with which it can make an accurate classification.

4.1.5 Challenges

Solving the important and open ABC problem is challenging for the three following reasons:

- *Small Signal-to-Noise Ratio.* Some regions in the timeline of long ITS are often far more relevant than others. For long series, the *signal-to-noise* ratio between the rel-

evant and irrelevant, or noisy, regions is often small. This makes finding signals challenging.

- *Unknown Signal Locations.* Relevant signals may occur anywhere in the continuous timeline. However, rarely are the *true* signal locations labeled, and so we assume no prior knowledge of which moments *should* be used for classification. Still, a good model must successfully find these discriminative moments, even without supervision.

- *Informative Irregularity.* Discriminative information often arises in the patterns of *when* observations are made. Capturing such *timing* is naturally different than modeling the dynamics of the values themselves and so a successful solution must consider both simultaneously.

4.1.6 Proposed Method: CAT

To address these challenges, we propose the **C**ontinuous-time **At**tention policy network (CAT), which solves the open Attention-based ITS Classification problem. CAT searches for relevant regions of input series via a reinforcement learning-based *Moment Network*, that learns to find *moments of interest* in the continuous timeline sequentially. At each predicted moment, our novel *Receptor Network* reads and represents the local temporal dynamics in the measurements along with any patterns that exist in the timing of observations through a continuous-time density function. Along the way, a recurrent *Transition Model* constructs a discriminative representation of the *transitions between moments of interest*, which is ultimately used to classify the series. CAT thus presents a novel paradigm for classifying ITS where intricate signals in long series are explicitly found and captured. Additionally, CAT generalizes recent ITS classifiers due to the flexibility of the *Receptor Network*, which can easily be augmented to leverage components of other recent ITS

models.

4.1.7 Contributions

Our contributions are as follows:

- We define the open problem of Attention-Based Classification (ABC) for irregular time series, which is common when learning from long and irregular inputs.
- Our solution, CAT, presents a novel framework that is the first to classify ITS by finding relevant moments in the *continuous* timeline, generalizing recent ITS classifiers.
- We identify and explore the weakness of state-of-the-art ITS classifiers when the signal-to-noise ratio is small.
- Experimentally, our approach successfully discovers signals in ITS and outperforms state-of-the-art classifiers on one synthetic and five real-world datasets.

4.2 Related Work

While our work is the first to consider the ABC problem for ITS, it is related to *ITS Classification* and *Input Attention*.

ITS Classification. Classifying irregular time series has recently become a popular and impactful problem as it generalizes many prior classification settings. To-date, most approaches [73, 138, 15] treat ITS classification as a *missing value imputation* problem: Create a set of evenly-spaced bins, then aggregate multiple values within each bin and estimate one value per empty bin. This estimation is a well-studied problem with a long history [102]. After imputation, regular time series classification may be performed. Some

recent ITS classifiers extend beyond simple imputation options (*e.g.*, mean) approaches by either including auxiliary information such as a *missingness-indicator* [73] or *time-since-last-observation* [15] as extra features to preserve properties found in the irregularity. Others build more complex value estimators by either learning generative models [70], using differentiable gaussian kernel adapters [107], or including decay mechanisms in Recurrent Neural Networks (RNN) to encode information-loss when variables go unobserved over long periods of time [89, 15]. Many works have also begun parameterizing ordinary differential equations to serve as time series models [61, 66, 101, 58]

Some recent models have begun to integrate attention mechanisms into ITS classification [108, 18, 115]. However, they still hand-pick reference timesteps for each input time series. Given long ITS with small signal-to-noise ratios, this decision is hugely impactful, as we show in our experiments. Moreover, by relying on RNNs for classification, these recent methods easily fail to capture signals when the number of estimated values gets too large. This requires the RNN to filter out many irrelevant timesteps in a long series, which is notoriously challenging due to both their slow inference and the vanishing gradient problem [51].

Input Attention. The goal of *Input Attention* is to discover relevant regions in the *input* space of a given instance and it has recently broken major ground in classifying images [85], graphs [67], text [111], and regularly-spaced time series [57, 96]. We refer to this as *input* attention, as such methods search for relevant regions in the *input space* of each instance. This approach is particularly impactful when inputs are high-dimensional as it explicitly disregards irrelevant regions of the input space.These methods also aid interpretability by clearly displaying which regions of an input were used to make a classification [9]. Input attention has yet to be considered for ITS despite strong implications of successful models.

This notion of attention differs from *attention mechanisms for recurrent neural networks*

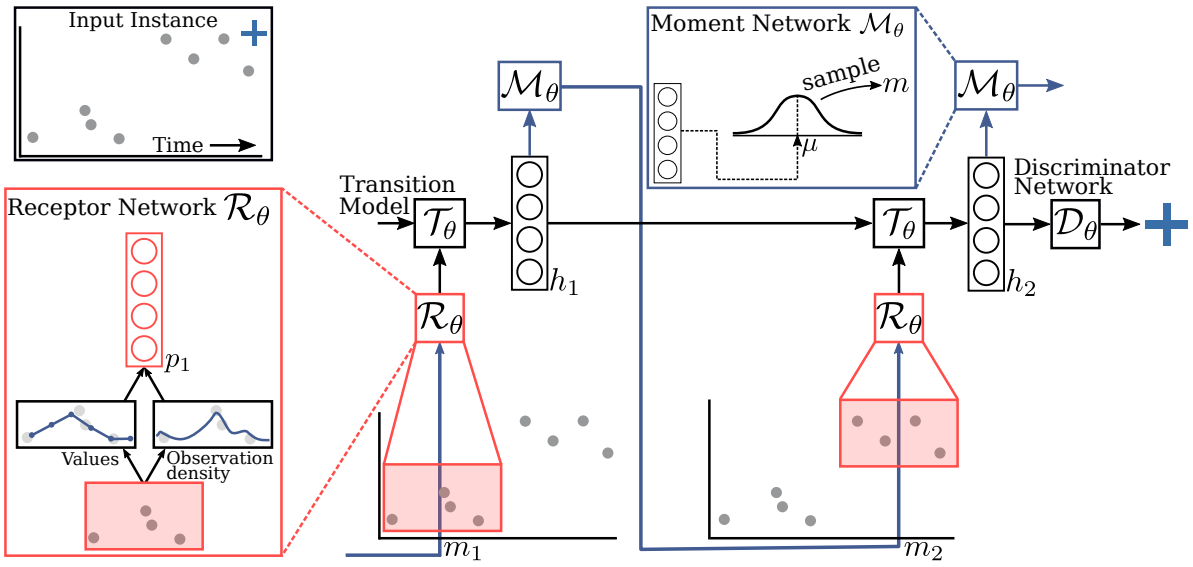


Figure 4.2: Overview of CAT. The *Receptor Network* models input values and irregularity around a moment m_i in the continuous timeline: $\hat{x}_i = \mathcal{R}(X, m)$. The *Transition Model* then updates its hidden state $h_i = \mathcal{T}(\hat{x}_i)$, modeling the transitions between *moments*. Then, the *Moment Network* parameterizes a Normal distribution from which it samples the next moment $m_{i+1} = \mathcal{M}(h_i)$. After iterating k times, the *Discriminator Network* predicts the final class: $y = \mathcal{D}(h_k)$, classifying the entire series.

[8], which learn attention distributions over the timesteps in the *latent* space of an RNN.

Local Discriminative Signals. Classifying time series by discovering locally discriminative subsequences, or shapelets [134], has been tremendously successful for particularly univariate and regularly-spaced time series [40]. While some works have extended shapelets into the multivariate setting [35, 68], to-date only preliminary work has been done for ITS [62]. Computing shapelets is notoriously slow, challenging to scale to multivariate signals, and has no clear and direct extension into the irregular setting.

4.3 Methods

4.3.1 Problem Definition

Given a set of N labeled irregular time series $\mathcal{D} = \{(X_i, y_i)\}_{i=0}^{N}$, consider the D variables of instance $X_i = [X_i^1, \ldots, X_i^D]$. To aid readability, all descriptions are provided in terms of one instance and one variable wherever possible. For each variable d, $X^d = [(t_1^d, v_1^d), \ldots, (t_{T^d}^d, v_{T^d}^d)]$, where t_i^d is the i-th timestamp of the d-th variable and v_i^d is its corresponding value. Timestamps t may differ between variables and the number of observations T^d may be unique to variable d. Additionally, we assume that the inputs X have low signal-to-noise ratios: The majority of the relevant information comes from a small proportion of the total timeline. There may still be multiple relevant regions, however. The goal is to learn a function $f : \mathbb{X} \rightarrow \mathcal{Y}$ that accurately maps input X to its accurate class y for previously-unseen time series. $\mathbb{X}$ is the input space of ITS and $\mathcal{Y} = \{0, \ldots, C\}$ is the set of C classes.

4.3.2 Proposed Method

An intuitive approach to solving the ABC problem is to first gather discriminative information from a region surrounding *one* randomly-initialized moment in the timeline of an ITS X. Then, use what is observed to choose a new moment from which to collect more knowledge. After k repetitions, the series can be classified based on all that was found.

We propose a novel architecture, the **C**ontinuous-time **Att**ention Policy Network (CAT), that encodes this key idea into four interdependent sub-networks that work in concert to classify long ITS: (1) A *Receptor Network* learns to model ITS observations (both the raw values *and* informative irregularity) local to a given *moment of interest*. (2) A *Transition Model* uses an RNN to represent the Receptor Network's findings with respect to the clas-

sification task. (3) A *Moment Network* learns to predict the next *moment of interest* given the Transition Model. A *moment of interest* is one point in the continuous timeline around which relevant information exists. After k repetitions, a (4) *Discriminator Network* classifies X based on the Transition Model's final representation. The *Receptor Network*, *Transition Model*, and *Discriminator Network* are trained together using standard supervised learning. Moments-of-interest are continuous so are sampled from a parameterized distribution in the *Moment Network* which is trained using reinforcement learning by rewarding accurate classifications. The architecture of CAT is illustrated in Figure 4.2.

Definition 4.3.1 (Smallest Greater and Largest Lesser)**.** Let t' be a real value and τ be a set of real-valued timestamps. A value $a \in \tau$ is the *Smallest Greater* (SG) if and only if there exists no other value $b \in \tau$ such that $b > t'$ and $b \leq a$. A value $a \in \tau$ is said to be the *Largest Lesser* (LL) if and only if there exists no other value $b \in \tau$ such that $b \leq t'$ and $a > b$. We refer to functions that return these values as $\text{SG}(t', \tau)$ and $\text{LL}(t', \tau)$, respectively.

Receptor Network

Given a moment of interest $m \in [0, \max T]$, our Receptor Network $\mathcal{R}_\theta$ predicts a vector representation the local *values* and *informative irregularity* within a width-δ window of X centered on moment m. $\max T$ is the largest timestamp in X. Thus $\mathcal{R}_\theta$ can be placed anywhere in the continuous timeline where it models local signals. To achieve this, we compute two w-dimensional vectors per variable: $\mathbf{p}$ represents X's values and $\mathbf{q}$ represents informative irregularity. For readability, we describe this process for just one variable and omit superscripts d since all variables are processed the same way and in parallel. First, all timestamps and values within this window are extracted into two vectors: τ contains the sequence of timestamps in the window $[m_i - \frac{\delta}{2}, m_i + \frac{\delta}{2}]$, and ν contains their corresponding values.

To compute $\mathbf{p}$, we linearly interpolate ν to estimate w values at a set of new timestamps within the window; The j-th element of vector $\mathbf{p}$ can be computed with respect a to timestamp $t' = m_i - \frac{\delta}{2} + \frac{j\delta}{w}$ for $j = [1, \ldots, w]$ as

$$\mathbf{p}_j = \frac{(\text{SG}(t', \tau) - t')\, \nu_{\text{LL}(t',\tau)} + (t' - \text{LL}(t', \tau))\, \nu_{\text{SG}(t',\tau)}}{\text{SG}(t', \tau) - \text{LL}(t', \tau)},$$

where $\text{LL}(t', \tau)$ is the largest timestamp less than t' and $\nu_{\text{LL}(t',\tau)}$ is its corresponding value. Similarly, $\text{SG}(t', \tau)$ is the smallest timestamp greater than t' and $\nu_{\text{SG}(t',\tau)}$ is its corresponding value. For the edge cases where a timestamp $t' > \max T$ or $t' < \min T$, the nearest value within the window is returned, flattening the edges of the interpolated window. If no observations occur in the entire window, an all-zero length-w vector is returned for the variable.

To compute $\mathbf{q}$, which represents *informative irregularity* within the window, we learn a function to represent the *timing* of observations, quantifying irregularity through the squared exponential function, inspired by [69]. Thus the j-th element of vector $\mathbf{q}$ as computed with respect to each $t' = m_i - \frac{\delta}{2} + \frac{j\delta}{w}$ for $j = [1, \ldots, w]$:

$$\mathbf{q}_j = \sum_{k=1}^{|\tau|} e^{-\alpha(t' - \tau_k)}, \tag{4.1}$$

where τ_k is the k-th element of sequence τ. Thus the *timing* of the observations is converted to a sequence of densities, which can often change between classes [73]. α controls the sensitivity of the kernel to the difference between t' and τ_k and can be hand-picked or learned during training [107].

Since the output of the Receptor Network will eventually inform the Moment Network in its prediction of the next moment m_{i+1}, we compute the representations for each variable at two granularities: One for capturing fine-grained local information, one for

capturing a coarse-grained representation of the entire series that is useful for both capturing long-term trends and for finding the next moments of interest, inspired by [85]. Once computed, we use a neural network to learn a L-dimensional representation $\hat{\mathbf{x}}_i$ for both $\mathbf{p}$ and $\mathbf{q}$, creating a vector representation of the width-δ window surrounding moment m_i:

$$\hat{\mathbf{x}}_i = \psi(\mathbf{W}[\mathrm{F}(\{\mathbf{p}^d\}_{d=1}^{D}), \mathrm{F}(\{\mathbf{q}^d\}_{d=1}^{D})] + \mathbf{b}), \tag{4.2}$$

where $\mathrm{F}(\cdot)$ and $[\cdot]$ denote flattening and concatenation, respectively. $\mathbf{W}$ and $\mathbf{b}$ are a matrix and vector of learnable parameters of shape $L \times 4w$ and $4w$, respectively. ψ is the rectified linear unit; L controls the representational capacity of the Receptor Network. To also incorporate *where* the collected data come from in the timeline, we concatenate m_i with $\hat{\mathbf{x}}_i$ before passing it to the Transition Model.

Transition Model

Next, a Transition Model $\mathcal{T}_\theta$ learns to represent the *transitions* between information gathered at each moment of interest. We follow the state-of-the-art for a vast array of sequential learning tasks [83] and implement this component as an RNN, creating one vector representation $\mathbf{h}_i$ per moment-of-interest. To avoid the vanishing gradient problem [51] pervasive to classic RNNs, we use a Gated Recurrent Unit (GRU) [19].

This recurrent component takes only k steps and k is typically kept very low ($k = 3$ in our experiments). In contrast, most recent models instead step through a large number of imputed timestamps T (typically $T \gg k$) creating slow models that are challenging to optimize.

Moment Network

The *Moment Network* $\mathcal{M}_\theta$ interprets the hidden state h_i of the *Transition Model* and predicts the next moment-of-interest m_{i+1}. There are no ground truth moments so we frame this component as a Partially-Observable Markov Decision Process (POMDP), similar to [85]. We follow the standard approach and solve this POMDP using on-policy reinforcement learning. In this way, the hidden state $\mathbf{h}_i$ from the *Transition Model* serves as an observation from the environment (representing the data collected at all prior moments of interest). The possible actions include all real-valued timestamps between 0 and $\max T$, and we define the reward to be the final classification success. The goal is to learn a policy $\pi(\mathbf{h}_i)$ that predicts the next moment m_{i+1}.

Since there are infinitely-many moments in the continuous timeline, we parameterize the mean μ_i of a Normal distribution with fixed variance from which we *sample* m_{i+1}. To achieve this, $\mathbf{h}_i$ is first projected into a one-dimensional probabilistic space by a neural network: $\mu_i = \sigma\left(\mathbf{W}\mathbf{h}_i + \mathbf{b}\right)$. The predicted μ_i is then scaled into the timeline via multiplication with $\max T$ and is used as the mean of a normal distribution from which we sample moment m_{i+1}. If $m_{i+1} > \max T$, we re-assign $m_{i+1} := \max T$, and similarly if $m_{i+1} < 0$, $m_{i+1} := 0$. This does not inhibit learning due to the REINFORCE algorithm discussed below. To train the Moment Network, we set reward $r_i = 1$ if the final classification is accurate and set $r_i = -1$ otherwise. The Moment Network thus seeks *discriminative* regions in the timeline which lead to the highest rewards.

CAT predicts k moments of interest, iteratively cycling between the Receptor Network, Moment Network, and Transition Model k times. After k steps, the final hidden state $\mathbf{h}_k$ represents all knowledge extracted from series X.

Discriminator Network

The final component of CAT is a *Discriminator Network* $\mathcal{D}_\theta$, which learns to project the Transition Model's final hidden state $\mathbf{h}_k$ into a C-dimensional probabilistic space in which it predicts its class probability $\hat{y}$. This final classification is made via a single linear layer: $\hat{y} = \text{softmax}(\mathbf{W}\mathbf{h}_k + \mathbf{b})$. This component is easily expandable according to the required complexity of a task.

Training the Continuous-Time Attention Policy Network

The Receptor Network, Transition Model, and Discriminator Network are optimized together to predict the class probability $\hat{y}$ as close to the true label y as possible. Since these networks are differentiable, we minimize the cross entropy:

$$\mathcal{L}_\text{s}(\theta_s) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})). \tag{4.3}$$

For readability, θ_s denotes all parameters of these networks that are optimized using standard supervised learning.

Predicted moments m are continuous so a key element of the Moment Network is *sampling* future moments. Instead, the learning objective of the Moment Network is the maximization of the expected reward: $R = \sum_{i=0}^{k} r_i$, so $\theta^*_\text{rl} = \arg\max_{\theta_\text{rl}} \mathbb{E}[R]$, where θ^*_rl is the optimal parameters for the *Moment Network*. This is not differentiable and cannot be optimized directly using back-propagation.

To make this optimization differentiable, we follow the standard protocol for on-policy reinforcement learning and optimize the Moment Network's policy using the REINFORCE algorithm [124]. Thus, we use a well-justified surrogate loss function that *is*

differentiable, allowing for optimization by taking steps in the direction of

$$\mathbb{E}[\nabla \log \pi(h_{0:k}, \mu_{0:k}, r_{0:k})R].$$

Thus the gradient can then be approximated for the predicted moments. This allows for learning but may also induce variance in the policy updates since this is not the *true* gradient for maximizing $\mathbb{E}[R]$. To reduce variance, we use a baseline b to approximate the expected reward, with which we may adjust the raw reward values, as shown in Equation 4.4. Here, b_j is a baseline predicted by a lightweight neural network and is encouraged to approximate the mean R via the reduction of their mean squared error. The weights θ_{rl} are thus updated with respect to how much better than average are the outcomes.

$$\mathcal{L}_{\text{rl}}(\theta_{\text{rl}}) = -\mathbb{E}\Bigg[\sum_{i=0}^{k} \log \pi(m_i|h_i)\Big[\sum_{j=i}^{k}\big(R - b_j\big)\Big]\Bigg] \tag{4.4}$$

Finally, the entire network can be optimized jointly and simultaneously via gradient descent of the loss function

$$\mathcal{L}(\theta) = \mathcal{L}_s(\theta_{\text{s}}) + \lambda\mathcal{L}_{\text{rl}}(\theta_{\text{rl}}), \tag{4.5}$$

where λ is used a hyperparameter that balances the scale of the loss terms and θ denotes all trainable parameters of CAT.

4.4 Experiments

4.4.1 Datasets

We evaluate the performance of CAT using one synthetic dataset and five real-world publicly-available datasets.

MΠ: We develop a synthetic binary classification dataset to demonstrate that CAT indeed finds intricate signals in long ITS data. To add signals for different classes, we center a width-Δ discriminative region around a random moment in the timeline for each time series. The values for the timestamps within the width-Δ window take one of two forms, depending on the class. One class is characterized by the values $\{1, 1, 1\}$ ("Π"-shaped), and the other by the values $\{1, 0, 1\}$ ("M"-shaped). The timestamps corresponding to these values are evenly-spaced in the width-Δ window. All timestamps *not* in the discriminative region are sampled uniformly across the timeline and values are sampled from a Normal distribution $\mathcal{N}(0, 1)$. In selecting Δ, we determine the signal-to-noise ratio of the data: A small Δ means that the "Π" or "M" signals happen in a short period of time, so overlooking the signal is punished more. We generate 5000 time series instances, each with 500 timestamps, and have an equal number of instances for each class.

UWAVE [76]: The popular UWAVE dataset contains 4478 length-945 gesture pattern time series collected from a handheld device. Each series is a member of one of eight classes. We follow the preprocessing procedure outlined by [69], randomly downsampling to 10% of the original values to create irregularity.

EXTRASENSORY [118]: We contribute a new approach to augmenting existing human activity recognition data by simulating listening probes, the full generation process for which is included in our publicly-available code. Listening probes monitor the activity of devices, collecting data only when certain conditions are met. While they save energy on edge devices, listening probes produce irregular time series. For example, consider detecting hand tremors from smartphones for digital health [34]. A listening probe on the accelerometer will collect data only when the phone moves rapidly, capturing hand tremors while the phone is carried. However, whenever the phone is set down or dropped, data are *also* collected, resulting in many irrelevant regions. Here, we consider a listening probe on a smartphone's *accelerometer* and detect common human activities.

Using the challenging EXTRASENSORY human activity recognition database, we extract four datasets collected via a listening probe on the 3-dimensional (x, y, and z axes) accelerometer records. When the norm of the difference between consecutive records surpasses a threshold $\gamma = 0.001$, the corresponding accelerometer data are collected. This differs from the random downsampling that prior works do as now irregularity can be informative. In our code, we provide further listening probes for the community.

We create four distinct datasets for detecting categories of human activity: WALKING (2636 time series), RUNNING (1066 time series), LYINGDOWN (7426 time series), and SLEEPING (9276 time series). For each class, we extract the smartphone data for the one person who performed the activity the most since activity patterns are often incomparable between people. We break each series into windows of 200 timestamps prior to applying the listening probe to ensure large-enough datasets and so the task is to detect whether or not the person performed the selected activity within this window. We finally balance each dataset to have an equal number of positive and negative series.

4.4.2 Compared Methods

We compare CAT to eight state-of-the-art ITS classifiers and one baseline version of CAT. The first four methods are simple imputation and feature expansion methods: linear interpolation: (RNN-interp), mean imputation (RNN-mean), mean imputation with the time-since-last-observation as an extra set of features (RNN-Δt), and mean imputation with a missingness indicator (RNN-S) [73]. The second group of four classifiers are state-of-the-art ITS classifiers: GRU-Decay [89], GRU-D [15], IPN [107], and mTAN [108]. We also compare against CAT-random, a random version of CAT. This ablates CAT by replacing the Moment Network with randomly-selected moments of interest.

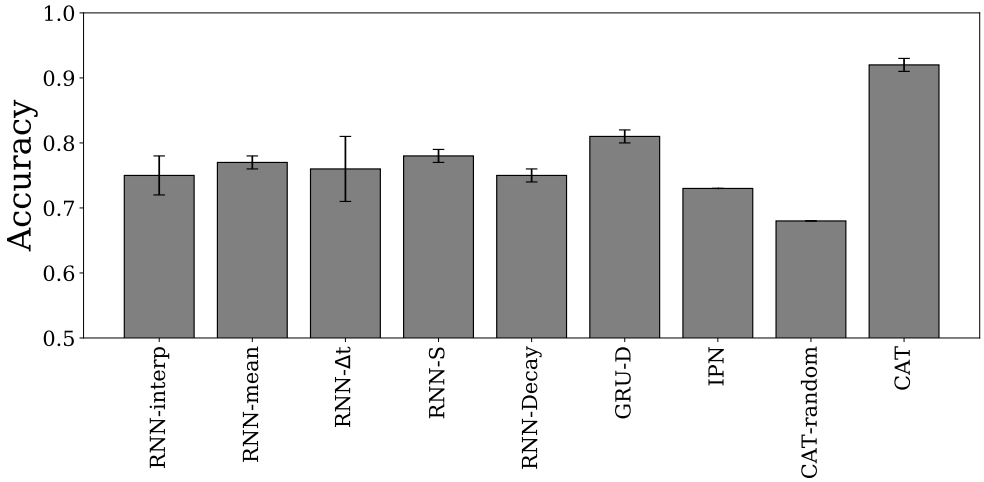


Figure 4.3: UWAVE classification with long series.

Datasets	Methods									
	RNN-interp	RNN-mean	RNN-Δt [73]	RNN-S [75]	RNN-Decay [89]	GRU-D [15]	IPN [107]	mTAN [108]	CAT-random	CAT (ours)
WALKING	0.65 (0.02)	0.64 (0.02)	0.64 (0.01)	0.64 (0.01)	0.64 (0.02)	0.65 (0.01)	0.65 (0.01)	0.68 (0.02)	0.60 (0.02)	**0.74 (0.03)**
RUNNING	0.52 (0.04)	0.52 (0.09)	0.52 (0.07)	0.46 (0.03)	0.44 (0.03)	0.43 (0.01)	0.46 (0.03)	**0.56 (0.01)**	0.47 (0.01)	**0.54 (0.01)**
LYINGDOWN	0.83 (0.05)	0.78 (0.04)	0.83 (0.04)	0.82 (0.08)	0.77 (0.04)	0.78 (0.03)	0.85 (0.04)	0.73 (0.00)	0.76 (0.01)	**0.87 (0.04)**
SLEEPING	0.76 (0.05)	0.79 (0.01)	0.79 (0.03)	0.76 (0.03)	0.78 (0.03)	0.76 (0.03)	0.77 (0.03)	0.76 (0.02)	0.73 (0.01)	**0.91 (0.02)**

Table 4.1: Testing accuracy of all methods with *infrequently*-imputed values for of the Human Activity Recognition Datasets. Parentheses denote standard deviations over five random train/test splits.

4.4.3 Implementation Details

For the UWAVE dataset, we use a standard 80% training, 10% validation, and 10% testing split. The EXTRASENSORY datasets contain instances taken from different windows along a single timeline. To avoid cross-contamination between training and testing data, we split instances *in time*, aiming for 80% training and 20% testing splits. The training/testing process is repeated five times and we report the average and standard deviation for all experiments. All methods use 64-dimensional hidden states for their respective RNNs. For CAT, we set the number of steps $k = 3$, use a 50-dimensional representation for the Receptor Network, and set $\alpha = 100$ in Equation 4.1 as determined using the validation data. All RNNs use GRU [19] recurrence equations.

Datasets	Methods									
	RNN-interp	RNN-mean	RNN-Δt [73]	RNN-S [75]	RNN-Decay [89]	GRU-D [15]	IPN [107]	mTAN [108]	CAT-random	CAT (ours)
WALKING	0.62 (0.04)	0.59 (0.02)	0.56 (0.02)	0.61 (0.04)	0.59 (0.02)	0.63 (0.02)	0.62 (0.01)	0.64 (0.01)	0.61 (0.01)	**0.77 (0.03)**
RUNNING	0.52 (0.05)	0.51 (0.09)	0.48 (0.02)	0.51 (0.07)	0.51 (0.05)	0.45 (0.01)	0.46 (0.01)	**0.55 (0.01)**	0.47 (0.00)	**0.54 (0.01)**
LYINGDOWN	0.78 (0.07)	0.8 (0.04)	0.83 (0.04)	**0.89 (0.02)**	0.88 (0.05)	0.82 (0.03)	0.85 (0.05)	0.74 (0.02)	0.77 (0.00)	**0.89 (0.03)**
SLEEPING	0.76 (0.05)	0.79 (0.02)	0.80 (0.03)	0.80 (0.02)	0.77 (0.02)	0.73 (0.01)	0.78 (0.04)	0.76 (0.02)	0.74 (0.01)	**0.90 (0.01)**

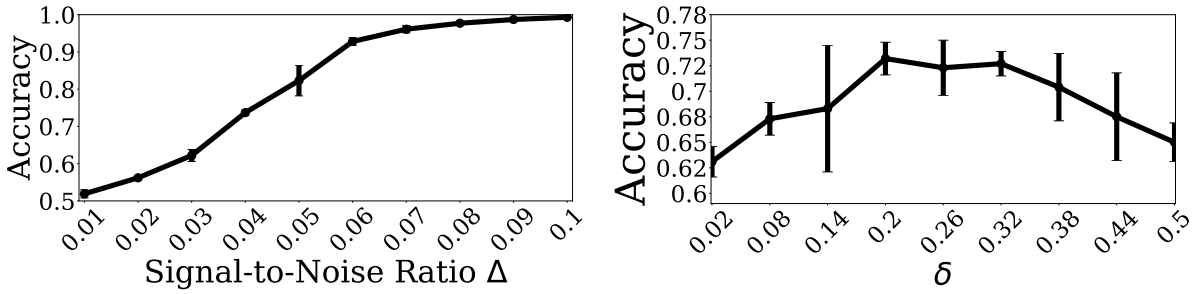
Table 4.2: Testing accuracy of all methods with *frequently*-imputed values for of the Human Activity Recognition Datasets. Parentheses denote standard deviations over five random train/test splits.

4.4.4 Experimental Results

Experiments on Real-World Data.

First, we demonstrate that CAT indeed handles long series better than the state-of-the-art methods. To achieve this, we impute the UWAVE data with 200 timestamps, which is much higher than prior experiments [107]. For ease of comparison, we also have CAT observe the data at the same "resolution" by setting $w = \delta * 200$ where δ is the receptor-width hyperparameter. This resolution can be tuned within CAT. Our results are reported in Figure 4.3. As expected, CAT achieves state-of-the-art accuracy on these data while the compared methods underperform their accuracy with roughly 100 imputed values. This indicates that CAT is far more robust to longer series than the state-of-the-art ITS classifiers.

Second, we show that CAT successfully captures *informative irregularity* in long series, as indicated by our results on the human activity recognition datasets (WALKING, RUNNING, LYINGDOWN, and SLEEPING). We compare all models using two settings: infrequent imputation (200 values) and frequent imputation (500 values). Intuitively, *frequent* imputation leads to clearer signals, as there are more values imputed on the signal, while *infrequent* imputation leads to unclear signals. To successfully classify these data, given infrequent imputation, finding the relevant regions of the data is more important. On the other hand, frequent imputations provide clear signals but come with the added



(a) Effect of signal width Δ.

(b) Effect of δ with $\Delta = 0.04$.

Figure 4.4: CAT's performance on Synthetic MΠ dataset.

risk of noise, requiring more explicit discovery of the relevant regions. Again to remain comparable to other methods, we set $w = \delta * 200$ and $w = \delta * 500$ for each respective frequency.

Our results for this experiment, shown in Tables 4.1 and 4.2, show that, as expected, CAT outperforms all compared methods in both the *infrequent* and the *frequent* settings for all datasets by an average of over 8%. The baselines also mainly perform their best with *infrequent* imputation, while CAT performs its best at *frequent* imputation as it adapts to different resolutions. Also as expected, the recent *GRU-D*, *IPN*, and *mTAN* models are generally CAT's strongest competitors. As expected, methods that model irregularity (*GRU-D*, *IPN*, *RNN-S*, and CAT) largely beat the methods that disregard irregularity. *RNN-interp*'s poor performance indicates that the benefits of CAT do not come from the linear interpolation used by the Receptor Network.

For all datasets, CAT outperforms *CAT-random*, the *policy-free* version of CAT that places the *Receptor Network* at random moments in the timeline. In fact, *CAT-random* is overall the *worst*-performing method, indicating that CAT's strong performance comes from a successfully-trained *Moment Network*. This demonstrates that CAT indeed succeeds to *learn* the discriminative regions of the given time series.

Experiments on Long Synthetic Data.

We finally evaluate the performance of CAT in a controlled setting using the MΠ dataset. With MΠ, we evaluate the robustness of our *Moment Network* and *Receptor Network* to changing the signal-to-noise ratio in the data and to changing how well the receptor width matches said signal-to-noise ratio. Each time series in MΠ is also quite long, having 500 timestamps, so for very small signal-to-noise ratios there is a massive proportion of noise with a very small relevant region. The results of these experiments are shown in Figure 4.4.

First, as shown in Figure 4.4a, we vary the signal-to-noise ratio in MΠ. Intuitively, as this ratio increases, the signal becomes easier for CAT to identify. By updating the *receptor width* δ to match the signal-to-noise ratio as it is increases, we find that the *Moment Network* indeed succeeds in finding the discriminative moments in the timeline, achieving nearly-perfect accuracy even with a signal-to-noise ratio as low as 0.06. Once the signal takes up 10% of the timeline, CAT consistently achieves 100% testing accuracy. We also find that the compared methods fail when the signal-to-noise ratio is lower than 0.1, achieving roughly 50% testing accuracy. This is not unexpected as RNNs are classically hard to train on such long series, especially with such noisy inputs.

Second, as shown in Figure 4.4b, we vary the *receptor width* parameter δ for a signal-to-noise ratio Δ of 0.04 to understand CAT's sensitivity to the proper selection of δ. We investigate the signal-to-noise ratio of 0.04 where CAT achieves only 75% accuracy, indicating potential sensitivity to hyperparameters (see Figure 4.4a). As expected, accuracy suffers both when δ is either too small (0.02) or too large (0.5). The optimal δ lies somewhere between 0.2 and 0.32 for this experiment. Quite interestingly, this is much larger than the data's signal-to-noise (0.04). While a larger receptor width δ should capture signals more easily, this suggests that the receptor still filters out the noisy regions when they

Dataset	δ	Irregularity
UWAVE	0.05	Off
Infrequent WALKING	0.2	Off
Infrequent RUNNING	0.05	On
Infrequent LYINGDOWN	0.05	Off
Infrequent SLEEPING	0.05	On
Frequent WALKING	0.2	Off
Frequent RUNNING	0.2	On
Frequent LYINGDOWN	0.2	On
Frequent SLEEPING	0.1	Off

Table 4.3: Best hyperparameter settings for CAT.

overlap with the receptor's window. These results also indicate that CAT can be robust to overestimating δ.

4.4.5 Hyperparameter Study

We experiment with three key hyperparameters of CAT for each dataset: The receptor-width δ, the hidden dimension of the Receptor Network $\mathcal{R}$, and whether or not to use the informative irregularity feature of CAT in the Receptor Network. Interestingly, we found that for $\mathcal{R}$, a hidden dimension of 50 seemed to consistently produce the best results. This hidden dimension largely controls the number of parameters in CAT and influences the timing experiments for which we also use a 50-dimensional representation. Our selections for δ values for different datasets are shown in Table **??**.

We tune δ between three values: 0.05, 0.1, and 0.2. For UWAVE, $\delta = 0.05$ was best. $\delta = 0.05$ was also best for all infrequent EXTRASENSORY datasets except for WALKING, which used 0.2. $\delta = 0.2$ was chosen for all frequent EXTRASENSORY datasets except for SLEEPING, for which $\delta = 0.1$.

For δ, we observe that for the *infrequent* experiments, a smaller receptor width is largely the best option while a larger width is beneficial for the *frequent* experiments. This

may be due to the fact that with the infrequent representation of the input series, closer focus on the comparatively-fuzzier signals is required. We also found that setting the number of steps $k = 3$ consistently outperformed larger and smaller values. While large values of k conceptually should still learn to classify effectively, in practice the more steps taken by a reinforcement learning agent per episode can make it more challenging to optimize effectively due to the credit assignment problem [113]. This fact may contribute to our finding $k = 3$ to be best.

We also find that there are cases where it is not essential to use both channels—Values and Irregularity—in the receptor network. While using the irregularity channel (computed via the squared exponential kernel in the main paper) always leads to state-of-the-art performance by CAT, its omission can sometimes improve CAT's performance slightly. When irregularity is an essential feature, however, this information cannot be removed. We show for which datasets this is true in Table **??**. This may be a feature of (1) how the irregularity is represented—there are other approaches—and (2) how essential it is to the task. We recommend always using the irregularity channel as the potential downside of ignoring irregularity outweighs the minor benefits of its omission in some cases.

4.4.6 Timing Experiments

Furthermore, CAT's *Transition Model* uses an RNN to model the transitions between *moments*, as opposed to the timestamps themselves. This hints that CAT should naturally be much faster than the compared methods. We confirm this by timing the training of all methods on the WALKING dataset with frequent imputation—see Figure 4.5. As expected, CAT runs over seven times faster than the next slowest method while achieving much higher testing accuracy. This is particularly meaningful for long series in time-sensitive domains such as healthcare where a model's inference time is hugely important

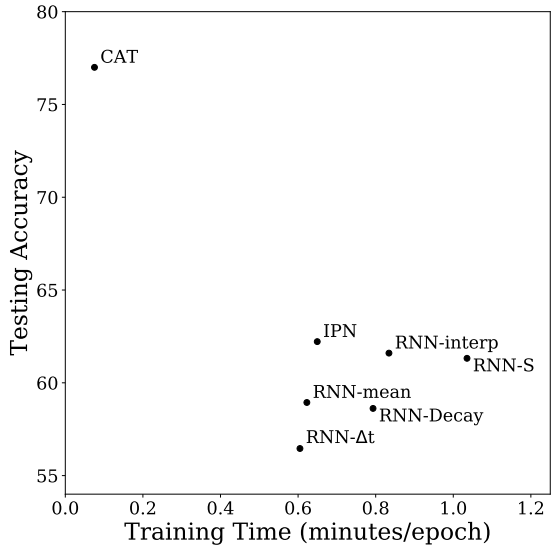


Figure 4.5: Timing performance for the high-resolution WALKING dataset. GRU-D takes over 3x longer than the next-slowest RNN-S and so is omitted from this figure.

[53]. Our reported timing comparisons between compared methods is also largely consistent with prior works' timing experiments [107]. Also as expected, the GRU-D [15] and mTAN [108] run significantly slower than the other compared methods and so omit their results from this figure. Their accuracies are much lower than CAT's—see Table 2 in the main paper. All models were trained and evaluated on Intel Xeon Gold 6148 CPUs.

4.5 Conclusions

In this work, we identify the open Attention-Based Classification problem for long and irregular time series, which is a challenging and impactful setting common to many important domains. The problem is to train a model to classify long irregular time series based on small discriminative signals in the continuous timeline while learning to ignore irrelevant regions. We then propose the Continuous-time Attention Policy Network (CAT) as its first solution. CAT learns to classify irregular time series by searching for often-small signals in a series' timeline. Our method includes a reinforcement learning-

based Moment Network that seeks discriminative moments in the timeline, positioning a novel Receptor Network that represents signals from *both* the values themselves and the patterns existing in the timing of the observations. Using a core Transition Model that learns to model the transition between moments, a Discriminator Network finally classifies the entire series. Across our experiments on one synthetic and five real-world datasets, CAT consistently outperforms eight state-of-the-art baselines by finding short discriminative signals.

5 Early Classification of Irregular Time Series

5.1 Introduction

5.1.1 Background

Early Classification is the task of predicting the class label of an ongoing time series before it is entirely observed [126]. As discussed in Chapter 1, Early Classification of Time Series is a core problem in time-sensitive domains where the earlier the prediction, the more time a user has to react.

However, in many time-sensitive domains, time series observations arrive *sporadically*, only recording *partial* information about an ongoing system. Such Irregular Time Series (ITS) thus have uneven gaps between their observations and are common to many impactful domains such as healthcare [73], climate science [7], and astronomy [100]. In a hospital, for instance, knowledge of a patient's health largely comes from doctors measuring a set of variables (*e.g.*, test results and vital sign recordings) over time. Within each variable, there are often irregular gaps between observations. Plus, multiple variables are rarely recorded at the exact same time, creating irregularity between variables. Learning to classify ITS *early* is an impactful and unexplored problem that can improve outcomes in time-sensitive domains.

5.1.2 State-of-the-art

There has recently been a surge of attention in both *early classification of time series* (ECTS) and classifying *irregular time series* (ITS), though to-date these fields have been developed separately.

To overcome the poor scalability and misclassification inherent to classic ECTS methods [125], recent works turn to reinforcement learning agents that predict whether to *Stop* or *Wait* at every step of an ongoing series [82, 46, 48]. This approach does not apply when time series are *irregular* because they disregard the amount of time that passes between observations. This manufactures late predictions as waiting until the next step will have a different cost depending on how long it is before the next observation. Existing multivariate methods also assume all variables are measured at every timestamp, which is rarely true for ITS. To-date, no method considers the irregularity of observations when deciding when to classify a series, a feature of the series that is crucial to capture [73].

Learning directly from ITS data is also an active research area. Most works use continuous time Recurrent Neural Networks (RNN) to represent observed values and/or irregularity of an ongoing series in their hidden states [15, 107, 73, 32]. Some works have recently extended to Neural Ordinary Differential Equation (ODE) models [101, 23, 55, 66, 61]. However, while these methods account for irregular sampling of a series' observations, they disregard the *earliness* of their predictions. They have yet to look beyond measuring only accuracy to consider more measures of practical usefulness.

5.1.3 Problem Definition

Our work is the first to consider Early Classification of Irregular Time Series (ECITS), which is an open, impactful, and challenging real-world problem. For a previously-unseen ITS X, we seek one small (early) real-valued time τ at which the entire series X may be classified accurately without using any observations later than τ. The goals of earliness and accuracy naturally conflict because early predictions are usually made at the expense of accuracy (fewer observations have been collected). This is naturally a multi-objective optimization problem.

5.1.4 Challenges

ECITS is challenging for three main reasons. First, earliness and accuracy conflict so a balance must be struck according to both the task at hand and each input instance. Second, the optimal halting time τ is unobserved and therefore unavailable for supervision or evaluation. Third, ITS are often sparse and multivariate. This is well-known to be challenging for machine learning where most successful models require fixed-length and regularly-spaced inputs.

5.1.5 Proposed Method: STOP&HOP

We overcome these challenges, proposing the first solution to the open ECITS problem which we refer to as STOP&HOP in reference to our key idea. STOP&HOP instantiates a general and modular solution to ECITS, integrating three essential components. First, an RNN-based *Prefix Encoder* embeds ITS data up to a candidate *halting time*, expanding on the recent success in representation learning for ITS. Then, a reinforcement learning-based *Halting Policy Network* decides whether or not to *Stop* and predict or *Wait* for more data. If it chooses to Wait, it also selects *for how long*, effectively *Hopping* some real-valued distance forward in time at which to reactivate the Halting Policy Network. Thus we formulate the early classification problem as a Partially-Observable Markov Decision Process with actions that operate on varying time scales. Thus the policy adapts to variations the irregularity of each individual input, using the timing of measurements to inform the earliness of predictions. Lastly, once the Halting Policy Network stops, a *Prefix Classifier Network* predicts the overall class label for the series. All components are trained jointly so that *earliness* and *accuracy* can be balanced easily.

5.1.6 Contributions.

This work's contributions are as follows:

1. We define the open problem of *Early Classification of Irregular Time Series* (ECITS), which bridges a major gap between modern early classification methods and real time-sensitive decision making problems.

2. Our proposed method, STOP&HOP, is the first solution to the ECITS problem, unifying the state-of-the-art for ITS and ECTS into one model.

3. We demonstrate that STOP&HOP learns to stop *exactly* when signals arrive using four synthetic datasets, leading to the earliest possible classifications. We also show that STOP&HOP succeeds to learn when to stop on three real-world time-sensitive tasks.

5.2 Related Work

Early Classification of Time Series. Early Classification of Time Series (ECTS) is the task of correctly predicting the label of a time series before it is fully observed. One early timestep is chosen per time series at which the whole instance is classified. While classifying sequences early is classically targeted only at time series [50, 128, 127, 37, 36, 87], some works have also extended to text [56] and video [78]. Most recent approaches [48, 46, 82, 25] have turned to deep learning, extending beyond traditional methods for univariate time series [88, 127, 126, 128], which scale poorly by exhaustively searching for discriminative subsequences [50]. The current state-of-the-art solution is to frame this problem as a Markov Decision Process (MDP), where at each regularly-spaced timestep a policy decides whether or not to stop and predict the label. Some use RNNs that halt

early [46] while others use Deep Q-Networks [82].

A major limitation of existing ECTS methods is their reliance on inputs being regularly-spaced as they decide whether or not to halt at each possible timestep. This does not account for missing values or gaps between observations, features essential to classifying ITS [15]. In ITS, the gaps between consecutive observations may even be large and unpredictable, so waiting until the next value arrives can have huge consequences.In the multivariate setting, multiple measurements are rarely taken concurrently, compounding this issue with existing works. Further, *the times at which observations arrive* can itself provide valuable knowledge for both earliness and accuracy [73]. A successful solution to our problem should take advantage of this extra source of information.

Learning from Irregular Time Series. Standard machine learning techniques often fail for ITS as they assume fixed-length and regularly-spaced inputs. To bridge this gap, there has been a lot of recent work to learn from ITS *directly*, developing models that take irregular series as inputs. Some approaches augment RNNs by either including auxiliary information such as a *missingness-indicator* [73] or *time-since-last-observation* [15] as extra features to preserve properties found in the irregularity. Others build more complex value estimators by either learning generative models [70], using gaussian kernel adapters [107, 69], set functions [54], or including decay mechanisms in Recurrent Neural Networks (RNN) to encode information-loss when variables go unobserved over long periods of time [89, 15]. Some recent works have begun parameterizing ordinary differential equations to serve as time series models [61, 66, 101, 58], Some very recent models have also begun to integrate attention mechanisms into this estimation process [108, 18, 116].

None of these existing ITS models consider *when* in the continuous timeline of an ongoing series they should return a prediction to the end user. A key constraint of the ECITS problem is that when classifying a series at a particular point in its timeline, *we cannot use any future values*. This hinders the direct use of any methods that interpolate

based on all observations, such at the Interpolation-Prediction Network [107] or ODE models that encode sequences backwards [101].

5.3 Methods

5.3.1 Problem Definition

Assume we are given a set of N labeled irregular time series $\mathcal{D} = \{(X^i, y^i)\}_{i=1}^N$. Each series X is a collection of one sequence of T^d (timestep, value) pairs per variable d: $X = \{\{(t_j^d, v_j^d)\}_{j=1}^{T^d}\}_{d=1}^D$ where each sequence of timesteps is strictly increasing ($t_1^d < t_2^d < \dots t_{T^d}^d$) and v_j^d is the corresponding value for each timestep. T^d denotes the number of observations for variable d. X is irregular in that typically $t_i^j \neq t_i^k$ and $t_{i+1}^j - t_i^j \neq t_{i+2}^j - t_{i+1}^j$ for all i, j, and k. Each label y indicates to which of C classes X belongs. Our goal of Early Classification of Irregular Time Series is to learn a function f that maps previously-unseen input time series to their accurate class labels y based *only* on values observed prior to some early time τ.The smaller τ is, the better. However, fully achieving both goals at the same time in practice is usually impossible since early predictions are often made at the expense of accuracy as less of the series has been observed. Thus we seek a tunable solution that balances *earliness* and *accuracy* according to the task at hand.

5.3.2 Proposed Method

We propose an intuitive first solution to the open Early Classification of Irregular Time Series (ECITS) problem, which we refer to as STOP&HOP and illustrate in Figure 5.1. The ultimate goal of our proposed method is to *predict the best halting time* τ for a given series so as to balance the cost of *delaying a prediction* with that of *misclassification*. Thus, one halting timestep τ is predicted for each series X along with a prediction $\hat{y}$ that is made

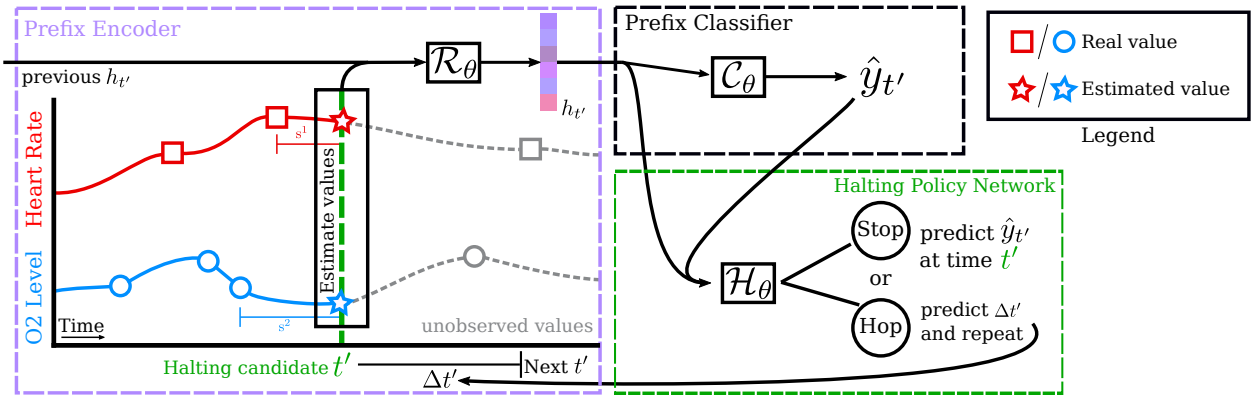


Figure 5.1: STOP&HOP Architecture. Given a timestamp t', an embedding is computed for all values and irregularity prior to t'. Then, a classifier attempts to classify the series. The halting policy network then uses this classification and the embedding to decide whether or not to stop or, if not, how long to wait before repeating this process.

using only observations made *before* time τ.

Since no solution to the ECITS problem exists, we begin by describing a general solution, which we then encode into a proposed architecture. A general solution to ECITS is the iteration of a three-step procedure: 1) Predict a *candidate* halting time t'. 2) Construct a vector representation h'_t of the ongoing series X that captures patterns in both values and irregularity up to time t'. 3) Predict whether or not to halt and classify X at time t'. If so, use $h_{t'}$ to classify X. If not, predict the next candidate halting time t'. Thus, a solution will march through the continuous timeline with stepsize dependent on the input data and at each step, will decide whether or not to Stop and return a classification. Each step of this general solution solves one problem of the ECITS problem. First, learning when to *try* to stop is essential in the irregular setting. This is in contrast to the standard ECTS setting where, with knowledge that observations arrive on a fixed schedule, methods simply decide whether or not to stop every time a new measurement arrives. Second, standard supervised learning methods struggle to model ITS data as they are not fixed-length. Learning dense representations of these data instead provides feature vectors that are easy to learn from. Step three can then leverage the vast success of deep learning to

classify ongoing ITS.

This modular approach solves the ECITS problem and so we instantiate this idea with solutions to each of the three open sub-problems. First, a continuous-time recurrent network $R(\cdot)$ constructs a representation $h_{t'} = R(X_{\leq t'})$ where $X_{\leq t'}$ represents all observations made prior to a timestep t'. Next, a *Halting Policy Network* decides either to *Stop* and predict, or *Hop* forward in time to a new timestep $t' := t' + \Delta t$ where Δt is a real-valued Hop Size computed as a function of representation $h_{t'}$. There is no analog in the ECTS literature which only choose between Stop and Wait for each observation. Since incoming observations are irregular, our adaptive approach instead allows the network to learn when to try and stop according to when observations arrive adding flexibility. During training, the halting policy network is encouraged to prefer both *smaller* (earlier) values $\hat{\tau}$ and *accurate* predictions. Once the Halting Policy Network chooses to stop or there are no more observations, a classifier network predicts the class label of X.

5.3.3 Prefix Embeddings for Irregular Time Series

STOP&HOP learns to produce embeddings of ongoing irregular time series via continuous-time representation learning, computing vector representations of a series X at real-valued timesteps. We refer to this as a *Prefix Encoder*, as it encodes the prefixes of ongoing time series. There has been a recent surge in approaches developed for representing ongoing ITS [15, 101] and most use a recurrent component to encode the series at real-valued timesteps in the continuous timeline: $h'_t = R(X, t')$, where $R(\cdot)$ is a continuous-time recurrent neural network and t' is a real-valued time. h'_t is thus a vector representing all dynamics of observations in series X prior to time t'. The only constraint on architecture design for $R(\cdot)$ in the Early Classification setting is that h'_t must only be computed with respect to values observed *earlier* than t'. This discourages the use of methods that com-

pute bi-directional hidden states or use future values for imputation [107]. Further, we seek to model the *irregularity itself*, which can inform both the classification accuracy and the earliness. Thus we compute $h_{t'}$ using the GRU-D [15], denoted to take variable step sizes between embeddings. The hidden state and input values are decayed based on the time since the previously-observed timestep for each variable as follows:

$$\hat{x}_{t'}^d = m_t^d x_t^d \odot (1 - m_{t'}^d)(\gamma_{x_{t'}}^d x_{\text{prev}}^d + (1 - \gamma_{x_{t'}}^d \tilde{x}_{t'}^d)) \tag{5.1}$$

$$\hat{h}_{t'} = \gamma_{h_t} \odot h_{t'} \tag{5.2}$$

$$r_{t'} = \sigma(W_r \hat{x}_{t'} + U_r \hat{h}_{t'} + b_r) \tag{5.3}$$

$$z_{t'} = \sigma(W_z \hat{x}_{t'} + U_z \hat{h}_{t'} + b_z) \tag{5.4}$$

$$\tilde{h}_{t'} = \phi(W \hat{x}_{t'} + U(r_{t'} \odot \hat{h}_{t'}) + V m_{t'} + b) \tag{5.5}$$

$$h_{t'} := (1 - z_{t'}) \odot h_{t'} + z_t \odot \tilde{h}_{t'}, \tag{5.6}$$

where $\tilde{x}_{t'}^d$ is the mean of all values of for the given instance's variable d before time t', $m_{t'}^d$ is a binary value indicating whether or not any new observations have been made since t', and x_{prev}^d is the value of the most recent observation of the d-th variable prior to time t'. $\odot$ is the hadamard product and $\gamma_{x_{t'}}^d$ is a decay factor for variable d at time t' computed by a neural network: $\gamma_{x_{t'}}^d = \exp(-\max(0, W_\gamma s_{t'}^d + b_\gamma)$ where $s_{t'}^d$ is the difference between t' and the time of the last observation of variable d.

An encoding $h_{t'}$ thus represents knowledge contained in the transitions between the values over time *and* the density of observations as measured by the indicator variable m used as input to the RNN. Measuring this density is important to both the classification and the earliness goals: Discriminative signals can appear in both the values themselves and their timing [73]. The density of measurements can thus indicate the likelihood of observing more relevant information in the near future [107]. In practice, these

update equations can be run multiple times *between* candidate halting times t' to ensure a granular-enough representation of a time series.

5.3.4 Classifying Prefixes

Given a prefix embedding $h_{t'}$, we use a *Prefix Classifier* $\mathcal{C}_\theta$ to predict the class label of the entire series X based only on values observed up to time t'. We use a standard fully connected network that projects $h_{t'}$ into a C-dimensional probabilistic space via the softmax function, generating the one probability for each of the C classes. In our experiments, we use only one hidden layer, but this component can be scaled up depending on the task at hand. Once the Halting Policy Network chooses to stop, the final prediction $\hat{y}$ for series X is also generated by the Prefix Classifier.

5.3.5 Halting Policy Network

STOP&HOP achieves early halting through a *Halting Policy Network* $\mathcal{H}_\theta$ that chooses whether to *Stop* or *Wait* at a given time t' based on the history of an ongoing time series X. Since there are rarely ground truth labels for when STOP&HOP *should* stop, we follow the state-of-the-art [46, 48, 82] and formulate this task as a Partially-Observable Markov Decision Process (POMDP): Given a state $h_{t'}$, select an action $a_{t'}$ from the set {Stop, Wait}. If $a_{t'} = \{\text{Stop}\}$, then the class label for the entire series is returned at time t'. Since X is irregular and observations may arrive sporadically, when the Halting Policy Network chooses to *Wait* it must also predict *for how long* before trying to stop again, thereby computing the next time t' at which the next prefix embedding can be computed. We refer to this as picking a *Hop Option*. Overall, the earlier and more accurate the predictions are for a series X, the higher the reward. If STOP&HOP stops too early, it may not have observed enough data and will be less likely to be accurate. Therefore, a good solution

must carefully balance between stopping and predicting the next halting time.

States. The Halting Policy Network's job is to compute a probability of halting $p_{t'}$ given the environment's state at time t' which can then be used to take an action: Stop now, or Wait for more data to be collected. While prior methods use only the prefix embedding $h_{t'}$ to represent the ongoing system's current state [46], we propose also predicting an intermediate classification $y_{t'}$ via the *Prefix Classifier* for each hidden state to gauge the model's current opinion of the classes. By feeding this information to the Halting Policy Network, it can learn the relationship between the Prefix Classifier's confidence and the likely accuracy, which often depends on the task. Additionally, we input t' as additional knowledge of a prediction's *earliness*.

Actions. The Halting Policy Network contains two prediction problems in sequence. First, it decides whether to *Stop* and classify, or to *Hop* forward in time and wait to classify. For the *Stop* decision, the model selects actions from the set $\mathcal{A}_{\text{stop}} = \{\text{Stop}, \text{Hop}\}$. We thus parameterize a binomial distribution over the action set $\mathcal{A}_{\text{stop}}$:

$$p_{t'} = \text{softmax}(W_h h_{t'} + U_h \hat{y}_{t'} + V_h t' + b_h), \tag{5.7}$$

where W, U, and V are learned weight matrices and b is a learned bias vector. Finally, we use the probabilities $p_{t'}$ to sample an action from a multinomial distribution. If $a_{t'} = \{\text{Stop}\}$, then the corresponding class prediction $\hat{y}_{t'}$ is returned at time t' and we set τ is set to t'.

If the model chooses to *Hop*, we run a hopping policy π_{hop}, another small neural network, that predicts a positive real-valued hop-size, which determines the next candidate halting time t'. Since we are working with the continuous timeline, this hop-time is naturally modeled using continuous values, which we sampled from a parameterized normal distribution. First, we parameterize the mean of this normal distribution using a neural

network:

$$\mu_{t'} = \sigma(Wh_{t'} + U\hat{y}_{t'} + Vt' + b), \tag{5.8}$$

then sample a hop-size Δt from the normal distribution with mean μ. We leave the standard deviation of this distribution as a hyperparameter, though in principle it can also be learned by the model. To ensure $\Delta t \geq 0$, we take the absolute value of Δt. In our experiments, we set the standard deviation to 0.1, indicating 10% of the timeline. Then, to compute the new candidate halting time, we add the hop-size to the current candidate halting time: $t' \mathrel{+}= \Delta t$.

In practice, early in training the *Halting Policy Network* may tend to exploit some actions by predicting unnecessarily high probability relative to those of an optimal policy. To encourage exploration early on in training, we employ a simple ϵ-greedy approach to action selection and exponentially decay the values ϵ from 1 to 0 throughout training:

$$a_{t'} = \begin{cases} a_{t'}, & \text{with probability } 1 - \epsilon \\ \text{random action}, & \text{with probability } \epsilon \end{cases}$$

This way, early on in training the model tries out different sequences of actions to cover the space of possible episodes more effectively. Furthermore, by increasing the probability that the Halting Policy Network does not stop early before the model has been thoroughly trained, the prefix encoder and discriminator also get to observe more of the sequences and increase their performance. Otherwise, a model that learns to stop early very quickly will never have seen the later portions of the training sequences. For all of our experiments, we compute ϵ as e^{-t} while consecutively reassigning $t = t + \frac{i*7}{E}$ for all E training epochs where t is initialized to 0. During testing, we set $\epsilon = 0$ so that there is no

random exploration for testing series.

Rewards. The final component of the POMDP is the *reward* for reaching different states. We encourage the halting policy network to cooperate with the prefix encoder and classifier by setting the reward $r_{t'} = 1$ when the $\hat{y}$ is accurate and setting $r_{t'} = -1$ otherwise.

5.3.6 Training

The Prefix Encoder and Classifier can be trained jointly using standard backpropagation, encouraging them to predict $\hat{y}$ as close to y as possible by minimizing their cross entropy:

$$\mathcal{L}_{\text{sl}} = \sum_{c=1}^{C} -(y^c \log(\hat{y}^c) + (1 - y^c) \log(1 - \hat{y}^c) \tag{5.9}$$

The Halting Policy Network, on the other hand, samples its actions and so its training is more intensive, though we follow the standard policy gradient method and use the REINFORCE algorithm [124] to estimate the gradient with which we update the network's parameters. To balance between earliness and accuracy, the parameters of the Halting Policy Network are updated with respect to two goals: Make τ small, and make $\hat{y}$ accurate. Following the state-of-the-art [46], we achieve smooth optimization by rewarding accurate predictions and penalizing *the cumulative probability of Waiting* over each episode. Thus the loss function for optimizing the halting policy network is computed as:

$$\begin{aligned}
\mathcal{L}_{\text{hpn}} = &- \mathbb{E}\left[\sum_{t'} \log \pi_{\text{stop}}(a_{t'}|h_{t'}) \left[\sum_{j=t'} (r_{t'} - b_{t'})\right]\right] \\
&- \mathbb{E}\left[\sum_{t'} \log \pi_{\text{hop}}(a_{t'}|h_{t'}) \left[\sum_{j=t'} (r_{t'} - b_{t'})\right]\right] \\
&- \lambda \sum_{t'} \log \pi(a_{t'} = \text{Stop}|h_{t'}),
\end{aligned}$$

where the scale of λ determines the emphasis on earliness. If λ is large, the halting policy network will learn to maximize the probability of stopping always whereas if λ is small or zero, the model will solely maximize accuracy. Interestingly, the most-accurate classification may not always be achieved by observing the entire series. For example, early signals followed by irrelevant values make classification challenging for memory-based models. Our approach deals with this case very naturally by learning not to make late predictions when they are less accurate, regardless of any cost of delaying predictions. The final loss is thus computed as $\mathcal{L} = \mathcal{L}_{\text{sl}} + \alpha\mathcal{L}_{\text{hpn}}$ and is minimized via gradient descent where α scales the loss components.

5.4 Experiments

5.4.1 Datasets

We evaluate STOP&HOP on four synthetic and three real-world datasets, which are described as follows:

Synthetic datasets: Since the true halting times are not available for real data, we develop four synthetic datasets with known halting times. Intuitively, a good early classifier will stop as soon as a signal is observed. To generate these data, we first uniformly sample $T-1$ timesteps from the range $[0, 1]$ for each of N time series. We set $T = 10$ and $N = 5000$. Then, from a chosen distribution of signal times, we sample one more timestep per series at which the class signal occurs and add it to a series' timesteps.

We experiment with four distributions of true signal times, each creating a unique dataset: UNIFORM $\mathcal{U}(0, 1)$, EARLY $\mathcal{N}(\mu = 0.25, \sigma = 0.1)$, LATE $\mathcal{N}(\mu = 0.75, \sigma = 0.1)$, and BIMODAL where half the signals are from EARLY and the other half are from LATE. For each time series, one value is sampled from one of the distributions—each distribution

creates one dataset—which serves as the time at which a signal arrives in the timeline. In all cases, we clamp this value to be within range $[0, 1]$. We generate two classes by giving 2500 time series a 1 at their signal occurrence time, and -1 to the remaining 2500 series in the dataset. This way, we know precisely when the signals arrive for each instance. Values corresponding to off-signal timesteps are set to 0 and are uniformly sampled from the timeline.

EXTRASENSORY: We use the publicly-available ExtraSensory [118] human activity recognition dataset, which contains smartphone sensor data collected across one week while participants labeled which actions they performed and when. Using these data, we simulate a listening probe on a smartphone's accelerometer data, which consist of three variables corresponding to the X, Y, and Z coordinates. A listening probe saves a phone's battery by collecting data only when certain measurements are taken, naturally creating irregular time series. For this dataset, we measure the norm of the 3-dimensional accelerometer data, only taking measurements associated with changes in the norm over 0.001. Here we consider the popular tasks of detecting WALKING and RUNNING, classifying whether or not a person performs an activity within a window. Since activity records are often incomparable between people, we use the records from the person who walked or ran the most breaking their series into 100-minute long windows. This creates two independent datasets, one per user. We balance each dataset, resulting in 2636 ITS with an average of 99 observation per series for WALKING and 3000 ITS with on average 100 observations for RUNNING.

PHYSIONET: The PHYSIONET dataset [109] contains medical records collecting from the first 48 hours after 4000 patients were admitted to an intensive care unit and is publicly-available. There are 37 variables recorded at irregular times for each patient along with one label indicating if they perished. We thus train our classifiers to perform mortality prediction for previously-unseen patients. In these data, 13.8% of patients have positive

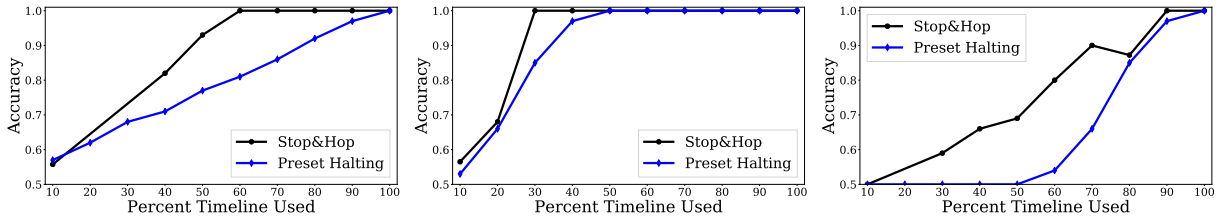
labels.

5.4.2 Implementation Details

For our synthetic datasets and PHYSIONET, we use a standard 80% training, 10% validation, and 10% testing split. The training set is used to learn model parameters and the validation set to evaluate the performance different hyperparameter settings. The testing set is used once to report the final evaluation metrics for each model. The EXTRASENSORY datasets contain instances taken from different windows along a single timeline and so we select a timestep for each before which is the training/validation data and after which is the testing. The training/testing process is repeated five times and we report the average and standard deviation for all experiments. For the SIMPLESIGNAL datasets, we use 10-dimensional hidden representations for the RNNs and the prefix encoder's hidden state is updated at intervals of 0.1. For EXTRASENSORY, a 20-dimensional prefix encoder's hidden state is updated every 10 minutes and for PHYSIONET it is updated every 4.8 hours. All models are optimized using Adam [63] with learning rates and weight decays chosen using the validation data.

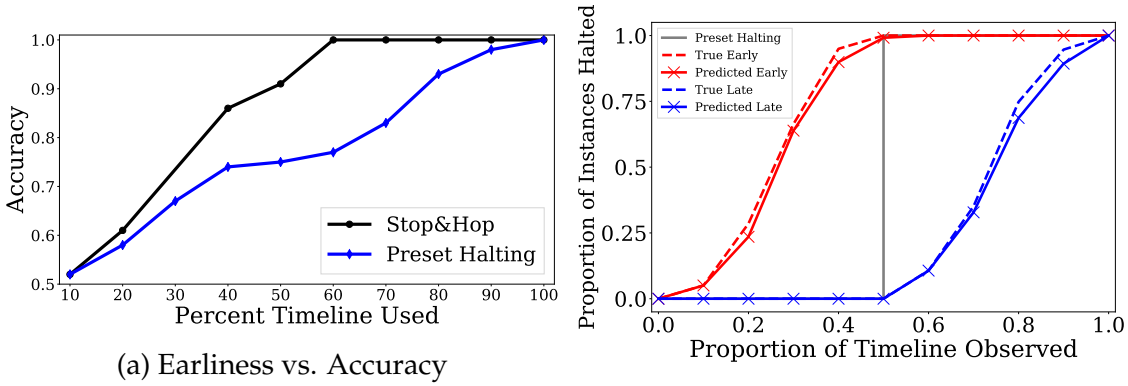
All models are implemented in Pytorch. Each model has a one-layer GRU-based continuous-time Recurrent Neural Network (RNN) that has 10 dimensions for the synthetic datasets and 20 dimensions for the real-world datasets. A one-layer neural network maps from the hidden states to the final classification.

For each method, we use a batch size of 32 and grid search for a learning rate (options: $\{10^{-2}, 10^{-3}, 10^{-4}\}$) and weight decay for L2 regularization (options: $\{10^{-3}, 10^{-4}, 10^{-5}\}$) using our validation data. The validation data is a random 10% of the training dataset and we repeat this random splitting five times.



(a) UNIFORM (b) EARLY (c) LATE

Figure 5.2: Earliness vs. Accuracy on three synthetic datasets—UNIFORM, EARLY, and LATE. The X axis denotes the *average* percent of the timeline used as STOP&HOP predicts one halting point per time series.



(a) Earliness vs. Accuracy

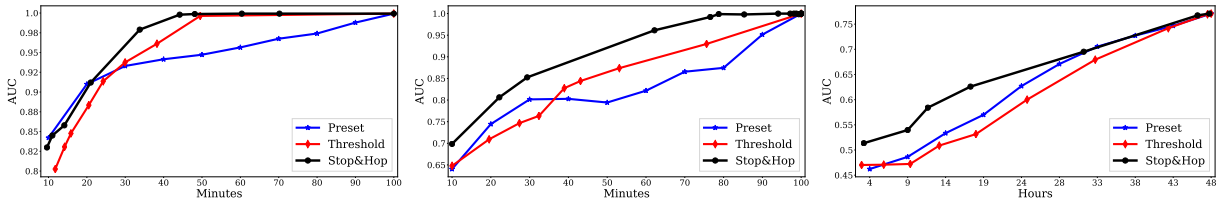
(b) Cumulative halting distribution

Figure 5.3: Results for synthetic BIMODAL dataset.

5.4.3 Synthetic Experiments.

We first verify that STOP&HOP indeed finds the true halting times by using use our four synthetic datasets where we know the halting times. Our results are shown in Figures 5.2 and 5.3, where we compare STOP&HOP to a *Preset Halting* baseline with the same Prefix Encoder as STOP&HOP in order to study the effects of *learning when to stop*. The preset halting method stops at a set of predetermined halting times. For example, a preset halting method that uses 50% of the timeline stops all instances at the same time.

To train STOP&HOP, we tune λ, the *wait penalty*, so that STOP&HOP halts on average at 10 to 100% of the timeline used and report the corresponding accuracy. We stop *Preset Halting* at the same points for comparison. To demonstrate that the halting policy network



(a) EXTRASENSORY WALKING (b) EXTRASENSORY RUNNING (c) PHYSIONET

Figure 5.4: Results on three real-world time-sensitive datasets.

learns an effective halting policy, we let it choose between stopping and hopping forwards with $\delta = 0.1$. One τ is predicted per series.

We find that STOP&HOP clearly achieves higher accuracy while using less of the timeline, as expected. This is only possible if STOP&HOP appropriately halts when it sees a signal and waits otherwise. As the four synthetic datasets have different halting distributions, we see that STOP&HOP succeeds to wait longer when signals are all later (Figure 5.2c) and stop earlier when signals are all earlier (Figure 5.2b).

For the BIMODAL dataset—see Figure 5.3—we first observe that STOP&HOP again makes early and accurate predictions, even when signals are distributed unevenly across the timeline. Further, we also show a snapshot of the halting distribution from the BIMODAL dataset from STOP&HOP trained with $\lambda = 3 \times 10^{-6}$ in Figure 5.3b. Each instance's predicted halting time is plotted against the proportion of the dataset with halting times earlier than a set of possible halting times, showing the cumulative halting distribution. We color-code the early and late signals and find that STOP&HOP matches the cumulative frequencies of the halting timings almost *perfectly*. As our method captures all positives exactly on time, matching the true cumulative functions without supervision, we postulate that STOP&HOP can also learn other complex functions. In contrast, the preset halting comparison's halting distribution is a step function: All instances halt at the same time. This is not flexible enough to match the halting distributions of real datasets.

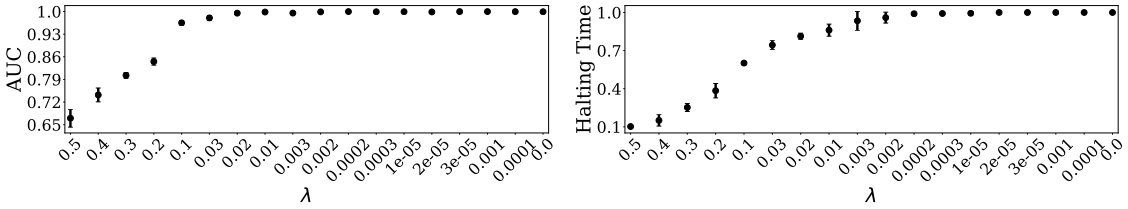
5.4.4 Real-world Dataset Experiments.

We further validate that STOP&HOP succeeds to find the appropriate halting times for ongoing ITS using three real-world datasets, our findings for which are shown in Figure 5.4. For these experiments, we add another compared method, a *Threshold Halting* approach that makes a prediction once its maximum predicted class probability surpasses a threshold. We range this threshold from 0.5 to 1.0 (wait until the end) and report the corresponding earliness and accuracy values. The same as *Preset Halting* and STOP&HOP, this method also uses a GRU-D to classify the series.

Optimal halting times are unknown for these datasets and so we plot the earliness and the accuracy of each method. In all cases, we find that STOP&HOP again clearly succeeds to make early and accurate predictions, even for real-world data. This is shown by STOP&HOP producing a curve that is higher and to the left of the other compared methods. As expected, signals may arrive at different times for different datasets. For EXTRASENSORY WALKING, STOP&HOP and *Threshold* are able to achieve perfect AUC using half the timeline. Again, as shown in Figure 5.3b, this happens when a method predicts good halting points. Our proposed method is consistently the best across the board.

The best results for STOP&HOP are obtained when the hop options are $\delta_1 = 5$, $\delta_2 = 10$, and $\delta_3 = 20$ minutes for the EXTRASENSORY datasets. For PHYSIONET, the best results are when $\delta_1 = 4.8$, $\delta_2 = 9.6$, and $\delta_3 = 14.4$ compared to each δ alone and to smaller or larger values. These specific δ values are chosen to break the full timeline—48 hours—into ten even steps.

We also conduct a hyperparameter study for λ, shown in Figure 5.5. Our results indicate that λ has strong control over the earliness–accuracy trade-off: As λ increases, the Halting Time and Accuracy steadily decrease. Standard deviations are computed across



(a) Effect of λ on AUC

(b) Effect of λ on Earliness

Figure 5.5: Hyperparameter tuning for λ, the emphasis on earliness, on the EXTRASENSORY RUNNING dataset.

five replications of the same experiment with different seeds. Small values indicate that STOP&HOP consistently learns a successful policy for these data.

5.4.5 Discrete Hop Size Experiments

We also consider selecting hop-sizes sample from a categorical distribution of hop options $[10, 20, 30]$ on the EXTRASENSORY RUNNING dataset. We train new STOP&HOP models with each possible combination of options, the results of which are shown in Figure 5.6. Here, we find that STOP&HOP is generally robust to the choice of hop options: Each set of options results in very similar curves. As shown in this experiment, however, this choice is the most conservative for STOP&HOP's performance—other option sets are even better in some cases. All hop options lead to performance that surpasses the compared methods.

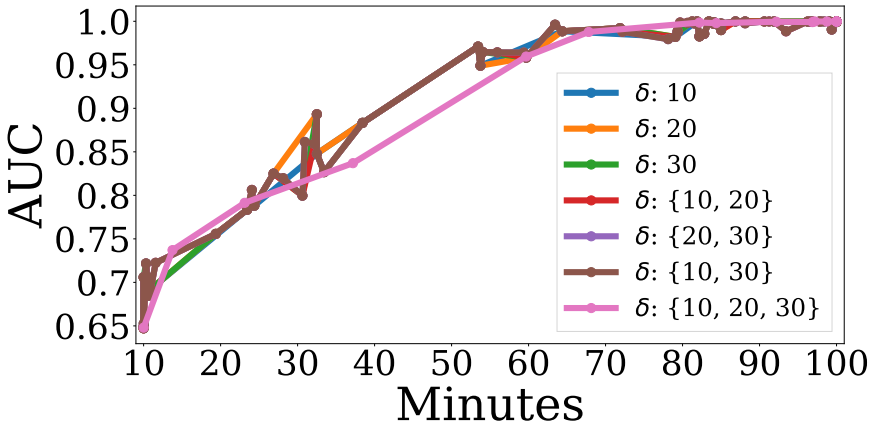


Figure 5.6: Comparing Hop Options on the EXTRASENSORY RUNNING dataset.

5.5 Conclusions

In this work we define the open Early Classification of Irregular Time Series problem and propose its first solution, STOP&HOP. This new problem is much more realistic than existing early classification settings. Our approach is a novel reinforcement learning-based continuous-time recurrent network that instantiates a modular and general framework to solve the ECITS problem. Our model learns to use patterns in both the values and irregularity of ongoing ITS to predict when enough data has been collected to warrant classification *with respect to the task at hand*. This is achieved without direct supervision on the halting times and our solution is naturally tunable between goals of *accuracy* and *earliness*. Using four synthetic datasets, we demonstrate that STOP&HOP halts at the earliest possible times. We then validate STOP&HOP on three real-world public datasets, demonstrating that our method outperforms existing methods in all settings, making early and accurate predictions.

Our work is also a step forward in ensuring the *practicality* of machine predictions. Accuracy (or a similar measure) is often used alone to evaluate a model. However, in time-sensitive domains, a classifier that learns to make earlier predictions and trade off some accuracy is preferred. There has been a large amount of recent work in many important directions—such as fairness and explainability—and we hope our work will also encourage the community to pursue measures of success beyond pure accuracy. In particular, the *timing* of a machine learning model's predictions can heavily impact their usefulness.

6 Conclusion

This dissertation studies and develops methods for learning the best time to classify data that are collected sequentially. Making such *timely* classifications is essential in a wide-variety of time-sensitive domains. We develop timely classifiers for both multi-class (Chapter 2) and multi-label (Chapter 3) classification of regularly-sampled time series. Then, we develop a method for learning from irregularly-sampled time series (Chapter 4) and finish with an early classifier for irregularly-spaced time series (Chapter 5), generalizing prior work in this space.

In the first part of this dissertation, we study the early classification of *regularly*-sampled time series. Chapter 2 introduces a deep reinforcement learning approach to solving this problem, pairing Recurrent Neural Networks with Policy Gradient methods to train adaptive models that predict when to classify ongoing time series. This approach is broadly applicable, as demonstrated in Chapter 4, and comprises a substantial impact to the field of early classification; this is now the state-of-the-art approach to this problem.

Chapter 3 extends early classification into the *multi-label* setting, where a given time series may be a member of multiple classes concurrently. This new problem setting introduces previously-unexplored challenges and opportunities: Evidence for different classes may (1) appear at different times in ongoing series, and (2) when class evidence appears sequentially, previously-observed classes can inform both future predicted classes *and* help predict them earlier. Our method leverages these insights, building on recent advances in classifier chains to train a successful early multi-label classifier. This approach generalizes prior work on early classification of time series, introducing a new notion of prediction timing.

In the second part of this dissertation, we relax the assumption that ongoing time se-

ries need to be regularly-sampled, instead allowing them to be sampled *irregularly*, where there are variable gaps between observations. Chapter 4 describes a novel method for explicitly finding discriminative regions in irregularly-sampled time series and using them to classify a series. These regions can be tiny relative to the entire timeline, and are challenging to detect using imputation methods. This approach is a novel paradigm for learning from irregularly-sampled time series, which generalizes prior methods and extends beyond the current literature. Further, this work introduces new notions of *ordered decomposition* of time series data into a sequence of subsequences: it can be easier to classify a time series by breaking it into windows in the correct order.

Chapter 5 ties early classification into the problem of learning from irregularly-sampled time series, generalizing both problems into one novel setting. Here, we present a new approach to learning from irregularly-sampled time series by learning *when in the timeline* to query a classifier, thereby allowing for early classifications. Additionally, previous early classifiers do not handle irregularly-sampled time series, and our approach paves the way for advancements in a more-general setting. This work is the first to combine prediction and observation timing into one, unified architecture, paving the way for mroe actionable and realistic early classification systems.

Altogether, the work presented in this dissertation expands the ways in which machine learning can support time-sensitive decision making problems. Our approaches move beyond measuring solely accuracy and consider the impacts of when a machine makes its predictions on desirable outcomes of a system. The next section discusses interesting directions for further advancements in this area.

7 Future Work

The work described in this dissertation is just the beginning of a plethora of fruitful research. There is a wide array of extensions of the work described in this document. This section describes three important areas of research that can follow the work presented in this dissertation.

7.1 Adaptive Intervention Lengths for Early Classification

In some time-sensitive problems, an expert may know how far in advance they would need to be warned of an impending event to take a rectifying action. For instance, it can take days to treat a dangerous case of sepsis; predictions made only a few hours before sepsis onset are less valuable than those made days in advance. This feature could be provided by our system. Early classification has not yet been studied in this "intervention-aware" context. To capture this, one might assume there exists some time-window δ that indicates *how far in advance of the event* a prediction must be made. This then imposes the constraint $\tau \leq T - \delta$, which is not easily satisfied because T is unknown at test time. The halting policy must then be developed to provide this functionality. While there are several avenues we will study to tackle this problem, the first attractive direction is to predict the *horizon*. Using the knowledge represented by the RNN one might predict $P(T|X)$, the time remaining before an event occurs. Then, the halting policy may be informed of the time-to-event (or even constrained), ensuring early-enough classifications given accurate horizon predictions.

7.2 Explaining Deep Early Classifiers

Using deep learning for early classification sacrifices the interpretability of classical approaches [125]. However, the burgeoning field of *explainability* is a promising direction for regaining this lost feature. Furthermore, complex models in time-sensitive decision making environments are often only be adopted if users can gain some intuitive understanding of a model's rationale to trust its predictions, regardless of their accuracy. For this, one might develop time-aware explanations of an early classifier's predictions, including the timing trade-offs, indicating the importance of timing versus feature content, and highlighting relationships between complex temporal features. Explainability for time series classification is itself a uniquely challenging setting as time series can be challenging to interpret visually. Therefore, the quality of explanation will likely depend on the given domain. For instance, the underlying shape of a time series may be shown to a practitioner, but other features such as trend or seasonality may be equally relevant in some settings. A successful explainable model will consider all possibly-relevant information, but intelligently procure only the most useful information.

7.3 Multivariate Convolutional Embeddings.

Some classification tasks can be solved by capturing stand-alone shapes, or *shapelets* [134], in ongoing time series values. While our general *Recurrent Halting Policy* approach can also capture shapes, in some cases we want this to be more explicit due to assumptions of a domain task or to enhance interpretability by matching to known domain-known shapelets. This can be achieved through the use of convolutional kernels, which have recently been shown to match well with shape-based time series classification [137, 139]. Integrating this powerful approach into the halting architecture remains entirely unstudied

in our challenging context. There is a broad avenue of possible formulations of Recurrent *Convolutional* Halting Policies, which can lead to a novel family of early classifiers. However, there remains a vast array of modern convolutional operators such as multi-channel approaches [137] or attention mechanisms [92], each of which will capture different assumptions. Such a convolutional approach may be developed to achieve intuitive and scalable multivariate early classification methods.

7.4 Early Multi-modal Classification.

Real time-sensitive domains often feature a plethora of data modalities, all recorded over time. For example, a hospital patient's vital signs are collected frequently with medical sensors while clinicians sporadically take notes on their observations. Some tests may require collecting various types of imaging and audio is beginning to be integrated into healthcare. Depending on the patient, there may also be medical imaging data collected over time. All data sources can contain important information, but they must be modeled quite differently. A clinical note, for instance, is unstructured, variable length, complex, and often extremely rich with expert insights. They may even contain rationales behind different treatment decisions. On the other hand, a sequence of test results are triggered by the healthcare worker's decision-making process. The resultant values are them the answers to an expert's queries.

Bibliography

[1] R. Adhikari and R. K. Agrawal. An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*, 2013.

[2] S. Aminikhanghahi and D. J. Cook. A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367, 2017.

[3] R. G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, and C. E. Elger. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. *Physical Review E*, 64(6):061907, 2001.

[4] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *ESANN*, 2013.

[5] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2017.

[6] M. T. Bahadori and Z. C. Lipton. Temporal-clustering invariance in irregular healthcare time series. *arXiv preprint arXiv:1904.12206*, 2019.

[7] M. T. Bahadori and Y. Liu. Granger causality analysis in irregular time series. In *SIAM International Conference on Data Mining*, pages 660–671. SIAM, 2012.

[8] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[9] Y. Bao, S. Chang, M. Yu, and R. Barzilay. Deriving machine attention from human rationales. In *Conference on Empirical Methods in Natural Language Processing*, pages 1903–1913, 2018.

[10] I. M. Baytas, C. Xiao, X. Zhang, F. Wang, A. K. Jain, and J. Zhou. Patient subtyping via time-aware lstm networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 65–74, 2017.

[11] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, arXiv, 2013.

[12] B. B. Biswal, M. Mennes, X.-N. Zuo, S. Gohel, C. Kelly, S. M. Smith, C. F. Beckmann, J. S. Adelstein, R. L. Buckner, S. Colcombe, et al. Toward discovery science of human brain function. *Proceedings of the National Academy of Sciences*, 107(10):4734–4739, 2010.

[13] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757 – 1771, 2004.

[14] J. S. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *NeurIPS*, 1990.

[15] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.

[16] J. R. Chen. Making subsequence time series clustering meaningful. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

[17] S.-F. Chen, Y.-C. Chen, C.-K. Yeh, and Y.-C. Wang. Order-free rnn with visual attention for multi-label classification. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[18] Y. Chen and J. Chien. Continuous-time attention for sequential learning. In *AAAI Conference on Artificial Intelligence*, 2021.

[19] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, 2014.

[20] D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification, arXiv, 2012.

[21] M. Coralie, G. Perrin, E. Ramasso, and R. Michèlle. A deep reinforcement learning approach for early classification of time series. In *EUSIPCO*, 2018.

[22] D. J. Daley and D. Vere-Jones. *An introduction to the theory of point processes: volume II: general theory and structure*. Springer Science & Business Media, 2007.

[23] E. De Brouwer, J. Simm, A. Arany, and Y. Moreau. Gru-ode-bayes: Continuous modeling of sporadically-observed time series. In *Advances in Neural Information Processing Systems*, pages 7379–7390, 2019.

[24] K. Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

[25] D. Dennis, C. Pabbaraju, H. V. Simhadri, and P. Jain. Multiple instance learning for efficient sequential data classification on resource-constrained devices. In *NeurIPS*, pages 10953–10964, 2018.

[26] A. F. Ebihara, T. Miyagawa, K. Sakurai, and H. Imaoka. Deep neural networks for the sequential probability ratio test on non-iid data series. *arXiv preprint arXiv:2006.05587*, 2020.

[27] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[28] P. Esling and C. Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):1–34, 2012.

[29] E. D. Feigelson, G. J. Babu, and G. A. Caceres. Autoregressive times series methods for time domain astronomy. *Frontiers in Physics*, 6:80, 2018.

[30] T.-c. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[31] Y. Fujita, N. Kanda, S. Horiguchi, K. Nagamatsu, and S. Watanabe. End-to-end neural speaker diarization with permutation-free objectives. In *Interspeech*, 2019.

[32] J. Futoma, S. Hariharan, and K. Heller. Learning to detect sepsis with a multitask gaussian process rnn classifier. In *International Conference on Machine Learning*, pages 1174–1182. JMLR. org, 2017.

[33] S. Gabarda and G. Cristóbal. Detection of events in seismic time series by time–frequency methods. *IET Signal Processing*, 4(4):413–420, 2010.

[34] I. García-Magariño, C. Medrano, I. Plaza, and B. Oliván. A smartphone-based system for detecting hand tremors in unconstrained environments. *Personal and Ubiquitous Computing*, 20(6):959–971, 2016.

[35] M. F. Ghalwash and Z. Obradovic. Early classification of multivariate temporal observations by extraction of interpretable shapelets. *BMC bioinformatics*, 13(1):1–12, 2012.

[36] M. F. Ghalwash, V. Radosavljevic, and Z. Obradovic. Extraction of interpretable multivariate patterns for early diagnostics. In *ICDM*, pages 201–210, 2013.

[37] M. F. Ghalwash, V. Radosavljevic, and Z. Obradovic. Utilizing temporal patterns for estimating uncertainty in interpretable early decision making. In *SIGKDD*, pages 402–411, 2014.

[38] M. Ghassemi, M. A. Pimentel, T. Naumann, T. Brennan, D. A. Clifton, P. Szolovits, and M. Feng. A multivariate timeseries modeling approach to severity of illness assessment and forecasting in icu with sparse, heterogeneous clinical data. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[39] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[40] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme. Learning time-series shapelets. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 392–401, 2014.

[41] A. Graves. Generating sequences with recurrent neural networks, arXiv, 2013.

[42] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE ICASSP*, pages 6645–6649, 2013.

[43] A. Gupta, H. P. Gupta, B. Biswas, and T. Dutta. An early classification approach for multivariate time series of on-vehicle sensors in transportation. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[44] T. Hartvigsen, W. Gerych, J. Thadajarassiri, X. Kong, and E. Rundensteiner. Learning to stop and classify ongoing irregular time series early. In *forthcoming*, 2022.

[45] T. Hartvigsen, C. Sen, S. Brownell, E. Teeple, X. Kong, and E. A. Rundensteiner. Early prediction of mrsa infections using electronic health records. In *HEALTHINF*, pages 156–167, 2018.

[46] T. Hartvigsen, C. Sen, X. Kong, and E. Rundensteiner. Adaptive-halting policy network for early classification. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 101–110. ACM, 2019.

[47] T. Hartvigsen, C. Sen, X. Kong, and E. Rundensteiner. Learning to selectively update state neurons in recurrent networks. In *CIKM*, 2020.

[48] T. Hartvigsen, C. Sen, X. Kong, and E. Rundensteiner. Recurrent halting chain for early multi-label classification. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2020.

[49] T. Hartvigsen, J. Thadajarassiri, X. Kong, and E. Rundensteiner. Continuous-time attention policy network for irregularly-sampled time series classification. In *forthcoming*, 2022.

[50] G. He, Y. Duan, R. Peng, X. Jing, T. Qian, and L. Wang. Early classification on multivariate time series. *Neurocomputing*, 149:777–787, 2015.

[51] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[52] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[53] S. Hong, Y. Xu, A. Khare, S. Priambada, K. Maher, A. Aljiffry, J. Sun, and A. Tumanov. Holmes: Health online model ensemble serving for deep learning models in intensive care units. In *SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1614–1624. ACM, 2020.

[54] M. Horn, M. Moor, C. Bock, B. Rieck, and K. Borgwardt. Set functions for time series. In *International Conference on Machine Learning*, pages 4353–4363. PMLR, 2020.

[55] Z. Huang, Y. Sun, and W. Wang. Learning continuous system dynamics from irregularly-sampled partial observations. In *Advances in Neural Information Processing Systems*, 2020.

[56] Z. Huang, Z. Ye, S. Li, and R. Pan. Length adaptive recurrent model for text classification. In *CIKM*, pages 1019–1027, 2017.

[57] A. A. Ismail, M. Gunady, L. Pessoa, H. C. Bravo, and S. Feizi. Input-cell attention reduces vanishing saliency of recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 10814–10824, 2019.

[58] J. Jia and A. R. Benson. Neural jump stochastic differential equations. In *Advances in Neural Information Processing Systems*, 2019.

[59] A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3, 2016.

[60] F. Kawala, A. Douzal-Chouakria, E. Gaussier, and E. Dimert. Prédictions d'activité dans les réseaux sociaux en ligne. In *4ième conférence sur les modèles et l'analyse des réseaux: Approches mathématiques et informatiques*, page 16, 2013.

[61] P. Kidger, J. Morrill, J. Foster, and T. Lyons. Neural controlled differential equations for irregular time series. In *Advances in Neural Information Processing Systems*, 2020.

[62] P. Kidger, J. Morrill, and T. Lyons. Generalised interpretable shapelets for irregular time series. *arXiv preprint arXiv:2005.13948*, 2020.

[63] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*. 2014.

[64] A. Konak, D. W. Coit, and A. E. Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, 91(9):992–1007, 2006.

[65] R. K. Lai, C.-Y. Fan, W.-H. Huang, and P.-C. Chang. Evolving and clustering fuzzy decision tree for financial time series data forecasting. *Expert Systems with Applications*, 36(2):3761–3773, 2009.

[66] M. Lechner and R. Hasani. Learning long-term dependencies inirregularly-sampled time series. In *Advances in Neural Information Processing Systems*, 2020.

[67] J. B. Lee, R. A. Rossi, S. Kim, N. K. Ahmed, and E. Koh. Attention models in graphs: A survey. *Transactions on Knowledge Discovery from Data*, 13(6):1–25, 2019.

[68] G. Li, B. Choi, J. Xu, S. S. Bhowmick, K.-P. Chun, and G. L. Wong. Shapenet: A shapelet-neural network approach for multivariate time series classification. In *AAAI Conference on Artificial Intelligence*, 2021.

[69] S. C.-X. Li and B. M. Marlin. A scalable end-to-end gaussian process adapter for irregularly sampled time series classification. In *Advances in Neural Information Processing Systems*, pages 1804–1812, 2016.

[70] S. C.-X. Li and B. M. Marlin. Learning from irregularly-sampled time series: a missing data perspective. In *International Conference on Machine Learning*, 2020.

[71] T. W. Liao. Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[72] Y.-F. Lin, H.-H. Chen, V. S. Tseng, and J. Pei. Reliable early classification on multivariate time series with numerical and categorical attributes. In *PAKDD*, pages 199–211, 2015.

[73] Z. C. Lipton, D. Kale, and R. Wetzel. Directly modeling missing data in sequences with rnns: Improved classification of clinical time series. In *Machine Learning for Healthcare*, pages 253–270, 2016.

[74] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzel. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

[75] Z. C. Lipton, D. C. Kale, and R. Wetzel. Modeling missing data in clinical time series with rnns. In *Machine Learning for Healthcare*, 2016.

[76] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan. uwave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[77] Z. Liu, L. Wu, and M. Hauskrecht. Modeling clinical time series using gaussian process sequences. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 623–631. SIAM, 2013.

[78] S. Ma, L. Sigal, and S. Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *CVPR*, pages 1942–1950, 2016.

[79] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth. Machine learning for internet of things data analysis: A survey. *Digital Communications and Networks*, 4(3):161–175, 2018.

[80] M. Marghany and M. Hashim. Retrieving seasonal sea surface salinity from modis satellite data using a box-jenkins algorithm. In *2011 IEEE International Geoscience and Remote Sensing Symposium*, pages 2017–2020. IEEE, 2011.

[81] R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[82] C. Martinez, E. Ramasso, G. Perrin, and M. Rombaut. Adaptive early classification of temporal sequences using deep reinforcement learning. *Knowledge-Based Systems*, 2019.

[83] G. Melis, T. Kočiskỳ, and P. Blunsom. Mogrifier lstm. In *International Conference on Learning Representations*, 2020.

[84] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *ISCA*, 2010.

[85] V. Mnih, N. Heess, A. Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

[86] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[87] U. Mori, A. Mendiburu, S. Dasgupta, and J. A. Lozano. Early classification of time series by simultaneously optimizing the accuracy and earliness. *IEEE transactions on neural networks and learning systems*, 29(10):4569 – 4578, 2018.

[88] U. Mori, A. Mendiburu, E. Keogh, and J. A. Lozano. Reliable early classification of time series based on discriminating the classes over time. *Data Mining and Knowledge Discovery*, 31(1):233–263, 2017.

[89] M. C. Mozer, D. Kazakov, and R. V. Lindsey. Discrete event, continuous time rnns. *arXiv preprint arXiv:1710.04110*, 2017.

[90] J. Nam, Y.-B. Kim, E. L. Mencia, S. Park, R. Sarikaya, and J. Fürnkranz. Learning context-dependent label permutations for multi-label classification. In *ICML*, pages 4733–4742, 2019.

[91] J. Nam, E. L. Mencía, H. J. Kim, and J. Fürnkranz. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *NeurIPS*, pages 5413–5423, 2017.

[92] M. Nauta, D. Bucur, and C. Seifert. Causal discovery with attention-based convolutional neural networks. *Machine Learning and Knowledge Extraction*, 1(1):312–340, 2019.

[93] H. F. Nweke, Y. W. Teh, M. A. Al-Garadi, and U. R. Alo. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications*, 105:233–261, 2018.

[94] J. Oh, J. Wang, and J. Wiens. Learning to exploit invariances in clinical time-series data using sequence transformer networks. In *Machine Learning for Healthcare Conference*, pages 332–347, 2018.

[95] Y.-S. Peng, K.-F. Tang, H.-T. Lin, and E. Chang. Refuel: Exploring sparse features in deep reinforcement learning for fast disease diagnosis. In *Neural information processing systems*, pages 7322–7331, 2018.

[96] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. W. Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2627–2633, 2017.

[97] N. F. M. Radzuan, Z. Othman, and A. A. Bakar. Uncertain time series in weather prediction. *Procedia Technol*, 11:557–64, 2013.

[98] A. Rajkomar, E. Oren, K. Chen, A. M. Dai, N. Hajaj, M. Hardt, P. J. Liu, X. Liu, J. Marcus, M. Sun, et al. Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine*, 1(1):18, 2018.

[99] H. Reuse, M. J. Joshi, R. Rascal, et al. Importance of data mining time series technique in crime and criminal investigation: A case study of pune rural police stations. *International Journal of Computer Applications*, 30(9), 2011.

[100] J. W. Richards, D. L. Starr, N. R. Butler, J. S. Bloom, J. M. Brewer, A. Crellin-Quick, J. Higgins, R. Kennedy, and M. Rischard. On machine-learned classification of variable stars with sparse and noisy time-series data. *The Astrophysical Journal*, 733(1):10, 2011.

[101] Y. Rubanova, T. Q. Chen, and D. K. Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, pages 5321–5331, 2019.

[102] D. B. Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

[103] J. Schmidhuber. Self-delimiting neural networks, arXiv, 2012.

[104] J. Schulman, N. Heess, T. Weber, and P. Abbeel. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, 2015.

[105] A. Sharma and S. K. Singh. Early classification of multivariate data by learning optimal decision rules. *Multimedia Tools and Applications*, pages 1–24, 2020.

[106] J. Shieh and E. Keogh. i sax: indexing and mining terabyte sized time series. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631, 2008.

[107] S. N. Shukla and B. Marlin. Interpolation-prediction networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2019.

[108] S. N. Shukla and B. M. Marlin. Multi-time attention networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2021.

[109] I. Silva, G. Moody, D. J. Scott, L. A. Celi, and R. G. Mark. Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. In *2012 Computing in Cardiology*, pages 245–248. IEEE, 2012.

[110] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[111] E. Sood, S. Tannert, P. Müller, and A. Bulling. Improving natural language processing tasks with human gaze-guided neural attention. In *Advances in Neural Information Processing Systems*, 2020.

[112] G. A. Susto, A. Cenedese, and M. Terzi. Time-series classification methods: Review and applications to power systems data. In *Big data application in power systems*, pages 179–220. Elsevier, 2018.

[113] R. S. Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 1984.

[114] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NeurIPS*, pages 1057–1063. 2000.

[115] Q. Tan, M. Ye, G. L.-H. Wong, and P. Yuen. Cooperative joint attentive network for patient outcome prediction on irregular multi-rate multivariate health data. In *International Joint Conference on Artificial Intelligence*, 2021.

[116] Q. Tan, M. Ye, B. Yang, S.-Q. Liu, and A. Ma. Data-gru: Dual-attention time-aware gated recurrent unit for irregular multivariate time series. In *AAAI Conference on Artificial Intelligence*, 2020.

[117] C.-P. Tsai and H.-Y. Lee. Order-free learning alleviating exposure bias in multi-label classification. In *AAAI*, 2020.

[118] Y. Vaizman, K. Ellis, and G. Lanckriet. Recognizing detailed human context in the wild from smartphones and smartwatches. *IEEE Pervasive Computing*, 16(4):62–74, 2017.

[119] O. Vinyals, S. Bengio, and M. Kudlur. Order matters: Sequence to sequence for sets. In *ICLR*, 2017.

[120] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 119:3–11, 2019.

[121] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu. Cnn-rnn: A unified framework for multi-label image classification. In *CVPR*, pages 2285–2294, 2016.

[122] M. Weber, M. Liwicki, D. Stricker, C. Scholzel, and S. Uchida. Lstm-based early recognition of motion patterns. In *ICPR*, pages 3552–3557. IEEE, 2014.

[123] J. Wiens, E. Horvitz, and J. V. Guttag. Patient risk stratification for hospital-associated c. diff as a time-series classification task. In *Advances in Neural Information Processing Systems*, pages 467–475, 2012.

[124] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[125] R. Wu, A. Der, and E. J. Keogh. When is early classification of time series meaningful? *arXiv preprint arXiv:2102.11487*, 2021.

[126] Z. Xing, J. Pei, and P. Yu. Early prediction on time series: A nearest neighbor approach. In *IJCAI*, pages 1297–1302, 2009.

[127] Z. Xing, J. Pei, and P. S. Yu. Early classification on time series. *Knowledge and Information Systems*, 31(1):105–127, 2012.

[128] Z. Xing, J. Pei, P. S. Yu, and K. Wang. Extracting interpretable features for early classification on time series. In *SDM*, pages 247–258, 2011.

[129] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 2048–2057, 2015.

[130] W. Yan, G. Li, Z. Wu, S. Wang, and P. S. Yu. Extracting diverse-shapelets for early classification on time series. *World Wide Web*, 2020.

[131] P. Yang, X. Sun, W. Li, S. Ma, W. Wu, and H. Wang. Sgm: Sequence generation model for multi-label classification. In *COLING*, pages 3915–3926, 2018.

[132] Q. Yang and X. Wu. 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, 5(04):597–604, 2006.

[133] L. Yao, E. Poblenz, D. Dagunts, B. Covington, D. Bernard, and K. Lyman. Learning to diagnose from scratch by exploiting dependencies among labels. *arXiv preprint arXiv:1710.10501*, 2017.

[134] L. Ye and E. Keogh. Time series shapelets: a new primitive for data mining. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 947–956. ACM, 2009.

[135] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *neural information processing systems*, pages 3391–3401, 2017.

[136] W. Zhang, D. K. Jha, E. Laftchiev, and D. Nikovski. Multi-label prediction in time series data using deep neural networks. *arXiv preprint*, abs/2001.10098, 2020.

[137] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, 28(1):162–169, 2017.

[138] K. Zheng, J. Gao, K. Y. Ngiam, B. C. Ooi, and W. L. J. Yip. Resolving the bias in electronic medical records. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2171–2180. ACM, 2017.

[139] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, pages 298–310. Springer, 2014.