Recurrent Halting Chain for Early Multi-label Classification

Thomas Hartvigsen
Worcester Polytechnic Institute
twhartvigsen@wpi.edu

Cansu Sen
Worcester Polytechnic Institute
csen@wpi.edu

Xiangnan Kong
Worcester Polytechnic Institute
xkong@wpi.edu

Elke Rundensteiner
Worcester Polytechnic Institute
rundenst@wpi.edu

ABSTRACT

Early multi-label classification of time series, the assignment of a label set to a time series before the series is entirely observed, is critical for time-sensitive domains such as healthcare. In such cases, waiting too long to classify can render predictions useless, regardless of their accuracy, while predicting prematurely can result in potentially costly erroneous results. When predicting multiple labels (for example, types of infections), dependencies between labels can be learned and leveraged to improve overall accuracy. Together, reliably predicting the correct label set of a time series *while* observing as few timesteps as possible is challenging because these goals are contradictory in that fewer timesteps often means worse accuracy To achieve early yet sufficiently accurate predictions, correlations between labels must be accounted for since direct evidence of some labels may only appear late in the series. We design an effective solution to this open problem, the Recurrent Halting Chain (RHC), that for the first time integrates key innovations in both Early and Multi-label Classification into one multi-objective model. RHC uses a recurrent neural network to jointly model raw time series as well as correlations between labels, resulting in a novel order-free classifier chain that tackles this time-sensitive multi-label learning task. Further, RHC employs a reinforcement learning-based halting network to decide at each timestep which, if any, classes should be predicted, learning to build the label set over time. Using two real-world time-sensitive datasets and popular multi-label metrics, we show that RHC outperforms recent alternatives by predicting more-accurate label sets earlier.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Supervised learning by classification**;

KEYWORDS

Early Classification; Multi-label Classification; Recurrent Neural Network; Reinforcement Learning

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403191

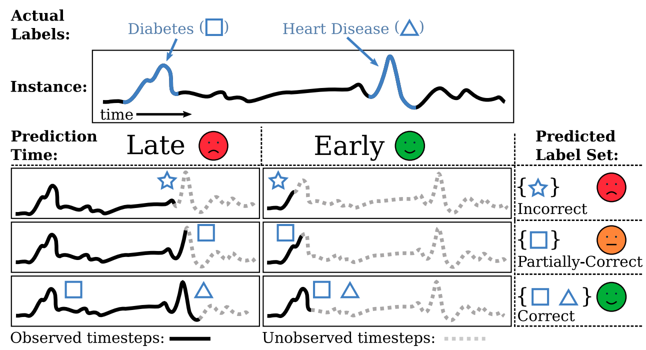


Figure 1: Early Multi-label Classification Problem. Multiple labels can be assigned to each instance of time series data. The correct labels must be predicted as early as possible. In this example, the bottom right box is ideal: The correct label set is predicted after observing very few timesteps.

ACM Reference Format:
Thomas Hartvigsen, Cansu Sen, Xiangnan Kong, and Elke Rundensteiner. 2020. Recurrent Halting Chain for Early Multi-label Classification. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, San Diego, CA, USA, August 4-8, 2020 , 11 pages. https://doi.org/10.1145/3394486.3403191

1 INTRODUCTION

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 [10] and transportation [9]. 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 [30]. We refer to the intersection of time-sensitive and multi-label learning as the Early Multi-label Classification (EMC) 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 the EMC problem 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 time clinicians have to react and intervene, thus improving patient outcomes. A concrete example of this setting is depicted in Figure 1 where the optimal outcome is achieved through the early and accurate prediction of both Diabetes and Heart Disease.

State-of-the-Art. Recently, major progress has been made in tackling the Early and Multi-label problems independently.

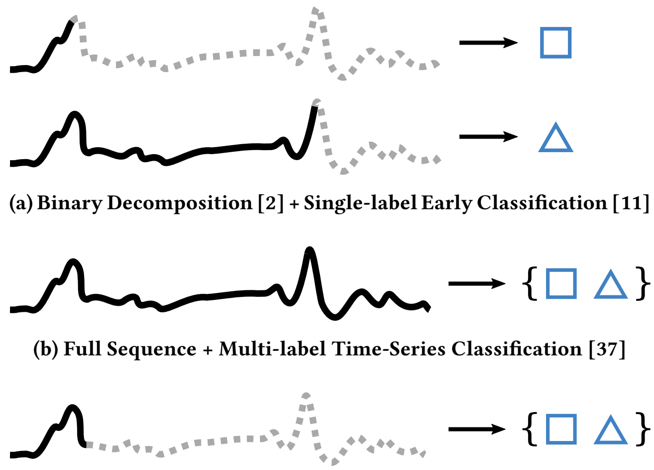
Early Classification has gained significant attention, particularly for applications using time series data [6, 11, 12, 16, 20, 33, 34], though initial work has also been done on text [14] and video [17]. Most recently, *tunable* Early Classification [11, 18, 20] has garnered much interest since the balance between *earliness* and *accuracy* tends to be task-dependent. For this reason, works have moved recently towards the use of neural networks for classification [4, 11, 14, 17, 18], seamlessly modeling high-dimensional inputs through recent advances in representation learning. 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 has also recently seen a surge of interest, in particular the study of *Classifier Chains* [3, 22, 23, 27, 30, 35, 36]. 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 [23, 30, 35] is that they require that a predefined label order be provided a-priori to enable optimization [3, 22, 27]. While this simplifies the problem, it unfortunately limits application to domains with easily-defined label orderings, such as speaker diarization [5]. 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.

Problem Definition. In this work, we are the first to address the open Early Multi-label Classification (EMC) problem, 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.

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 [2] + Single-label Early Classification [11]

(b) Full Sequence + Multi-label Time-Series Classification [37]

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

Figure 2: Comparing the different solutions to the early multi-label classification problem.

Proposed Method. We propose a solution to the new EMC 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 EMC 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 *Discriminator* 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 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.

Contributions. Our main contributions are summarized below:

- We define the new open problem of Early Multi-label Classification (EMC) with its roots in both Early Classification and Multi-label Classification.
- We design the first solution to EMC, 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.

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 [6–8, 12, 20, 32–34], however the most recent approaches [4, 11, 18] propose a general formulation of this problem through the use of neural networks. By using neural networks, these approaches naturally model multivariate inputs [11, 18] in contrast to previous works which solely study univariate inputs [6, 21, 32–34]. The univariate approaches typically involve exhaustive search for discriminative subsequences, which scales poorly into the multivariate setting [12]. Additionally, many recent works also take a *prefix-based* approach to early classification[11, 20, 21], learning at which timestep enough information has been observed to warrant classification. This is in contrast to *shapelets* [32], 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 [11, 18]. [11] 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. [18] proposes a Deep Q Network [19] 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 [25].

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.

Table 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 $X_{0,\cdots,t}^{(i)}$ from the RNN.
$\pi(\cdot)$	Policy – maps hidden states to actions.
τ^l	Predicted halting-step for class l.
	where $i = [1, \ldots, N]$, $t = [1, \ldots, T]$, and $l = [1, \ldots, L]$.

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 [2]. 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 [3, 22, 23, 27, 30, 35, 36]. 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 [3, 27]. Meanwhile, the chosen label order dramatically impacts the performance of the classifier chain [23, 29]. Recent works have just begun to remove this assumption, proposing classifier chains based on confidence-ranked labeling [3] and multi-task learning [27].

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.

3 METHODOLOGY

3.1 Problem Definition

Given a set of labeled time series containing N length-T 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

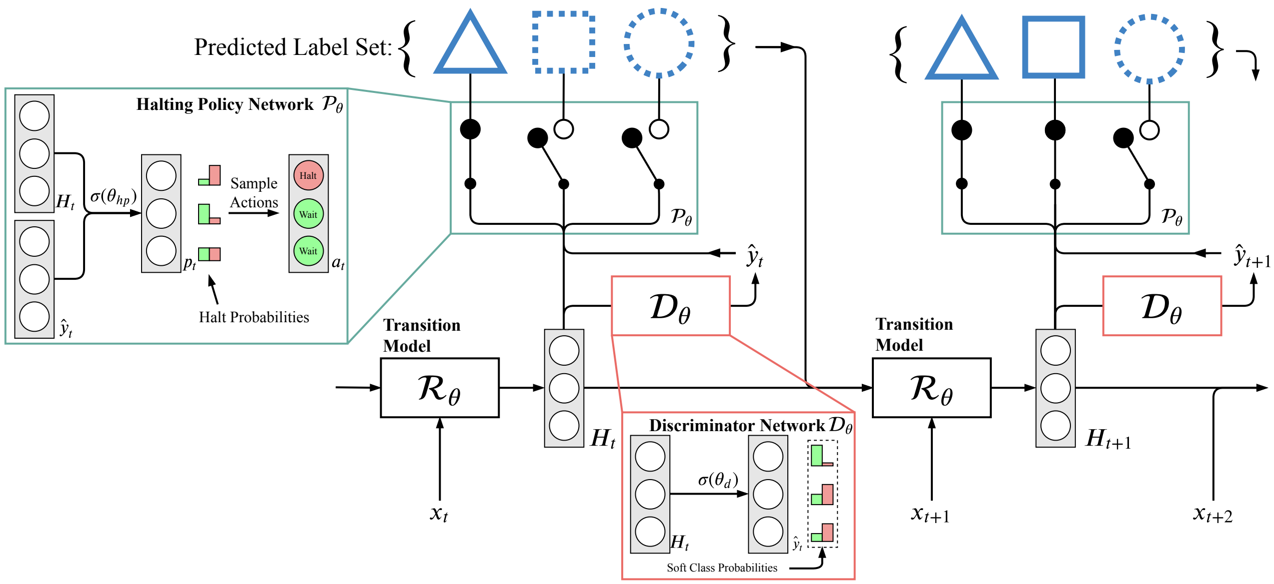


Figure 3: Overview of RHC. At each timestep t, the Transition Model $\mathcal{R}$ computes $H_t = \mathcal{R}(H_{t-1}, x_t, \bar{y}_{t-1})$, the new hidden state for timestep t. Using H_t, the Discriminator $\mathcal{D}$ computes soft class probabilities for every class: $\hat{y}_t = \mathcal{D}(H_t)$. 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.

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 $f_\theta(X) \longrightarrow Y$ 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 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.

3.2 Proposed Method: RHC

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 $X \longrightarrow Y$ 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 [3, 27] 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.

3.2.1 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) [13] cells as our transition function, mapping inputs x_t to a representation H_t as follows:

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

$$i_t = \sigma(W_i \cdot [H_{t-1}, X_t] + b_i) \tag{2}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \phi(W_c \cdot [H_{t-1}, X_t] + b_c) \tag{3}$$

$$o_t = \sigma(W_o \cdot [H_{t-1}, X_t] + b_o) \tag{4}$$

$$H_t = o_t \odot \phi(C_t) \tag{5}$$

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 [3]. 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 $\bar{y}_t$:

$$f_t = \sigma(W_f \cdot [H_{t-1}, X_t, \bar{y}_{t-1}] + b_f) \tag{6}$$

$$i_t = \sigma(W_i \cdot [H_{t-1}, X_t, \bar{y}_{t-1}] + b_i) \tag{7}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \phi(W_c \cdot [H_{t-1}, X_t, \bar{y}_{t-1}] + b_c) \tag{8}$$

$$o_t = \sigma(W_o \cdot [H_{t-1}, X_t, \bar{y}_{t-1}] + b_o) \tag{9}$$

$$H_t = o_t \odot \phi(C_t) \tag{10}$$

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.

3.2.2 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 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.

$$\begin{aligned} \hat{y}_t &= \mathrm{P}(Y \mid H_t) = \mathcal{D}_\theta(H_t) \\ &= \frac{1}{1 + e^{W_{ho}H_t + b_{ho}}} \end{aligned} \tag{11}$$

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.

3.2.3 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 [11, 18], 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 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.

$$p_t = \sigma(W_{hp}[H_t, \hat{y}_t] + b_{hp}) \tag{12}$$

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:

$$\bar{y}_t = \bar{y}_{t-1} + a_t \odot (1 - \bar{y}_{t-1}) \tag{13}$$

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 3.2.4, 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 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.

$$a_t = \begin{cases} a_t, & \text{with probability } 1-\epsilon \\ \text{random action}, & \text{with probability } \epsilon \end{cases} \tag{14}$$

3.2.4 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 15), using standard back-propagation since all operations are differentiable, similar to [3]. To achieve this, $\hat{y}^l$ is simply the Discriminator's prediction of class l *from the timestep at which it was predicted.*

$$\mathcal{L}_{\text{sl}}(\theta) = \sum_{l=1}^{L} -(y^l \log(\hat{y}^l) + (1-y^l)\log(1-\hat{y}^l)) \tag{15}$$

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 [31] as a gradient estimator to train the network. The learning objective of the halting policy network is the maximization of the expected return $R = \sum_{t=0}^{\tau} r_t$:

$$\theta^*_{hp} = \underset{\theta_{hp}}{\arg\max}\, \mathbb{E}[R] \tag{16}$$

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 [26]. The new objective can be optimized using gradient descent by taking steps in the direction of $\mathbb{E}[\nabla \log \pi(H_{0,\cdots,\tau}, a_{0,\ldots,\tau}, r_{0,\cdots,\tau})R]$ [24]. The gradient can then be approximated for the halting decisions for each class as shown in Equation 17. 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.

$$\mathcal{L}^l_{\text{rl}}(\theta) = -\mathbb{E}\left[\sum_{t=0}^{\tau^l} \log \pi(a^l_t|H_t)\left[\sum_{t'=t}^{\tau^l}(R^l - b^l_{t'})\right]\right] \tag{17}$$

where b^l_t 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 17 across all l classes, resulting in one final differentiable function summarizing the success of the halting policy network:

$$\mathcal{L}_{\text{rl}}(\theta) = \frac{1}{L}\sum_{l=0}^{L} \mathcal{L}^l_{\text{rl}}(\theta) \tag{18}$$

3.2.5 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 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.

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{sl}} + \mathcal{L}_{\text{rl}} + \lambda \sum_{l=0}^{L}\sum_{i=0}^{\tau^l} \log \pi(a^l_t = 1|H_t) \tag{19}$$

4 EXPERIMENTS

4.1 Datasets

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

`HAR` [1]: 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 label 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.

`ExtraSensory` [28]: 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.

Table 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".

Time-Steps Observed	Evaluation Metrics	LSTM-BD [17]	E-LSTM [4]	LSTM-CC [30]	EARLIEST [11]	RHC (ours)
20%	Instance-AUC↑	0.88 (0.00)	0.85 (0.00)	0.90 (0.02)	**0.92** (0.00)	**0.92** (0.01)
	Micro-AUC↑	0.86 (0.00)	0.85 (0.00)	0.88 (0.01)	**0.91** (0.00)	**0.91** (0.00)
	Macro-AUC↑	0.86 (0.00)	0.84 (0.00)	0.88 (0.02)	**0.91** (0.00)	**0.91** (0.00)
	Hamming Loss↓	0.18 (0.00)	0.21 (0.00)	0.17 (0.01)	**0.13** (0.00)	**0.13** (0.01)
	Micro-F1↑	0.62 (0.00)	0.57 (0.00)	0.66 (0.03)	**0.74** (0.00)	0.72 (0.02)
	Macro-F1↑	0.62 (0.00)	0.57 (0.00)	0.65 (0.03)	**0.74** (0.00)	0.71 (0.02)
40%	Instance-AUC↑	0.91 (0.00)	0.89 (0.00)	0.92 (0.02)	**0.94** (0.00)	**0.94** (0.00)
	Micro-AUC↑	0.90 (0.00)	0.90 (0.00)	0.92 (0.02)	0.92 (0.00)	**0.93** (0.00)
	Macro-AUC↑	0.91 (0.00)	0.89 (0.00)	0.92 (0.02)	0.93 (0.00)	**0.94** (0.00)
	Hamming Loss↓	0.17 (0.00)	0.17 (0.01)	0.15 (0.01)	**0.10** (0.00)	**0.10** (0.00)
	Micro-F1↑	0.65 (0.00)	0.67 (0.02)	0.72 (0.02)	0.79 (0.00)	**0.81** (0.00)
	Macro-F1↑	0.63 (0.00)	0.68 (0.02)	0.72 (0.02)	0.79 (0.00)	**0.81** (0.00)
60%	Instance-AUC↑	0.92 (0.00)	0.93 (0.01)	0.93 (0.01)	0.94 (0.00)	**0.95** (0.00)
	Micro-AUC↑	0.92 (0.00)	0.94 (0.01)	0.93 (0.01)	0.93 (0.00)	**0.95** (0.00)
	Macro-AUC↑	0.90 (0.00)	0.94 (0.01)	0.93 (0.01)	0.94 (0.00)	**0.95** (0.00)
	Hamming Loss↓	0.13 (0.00)	0.13 (0.02)	0.13 (0.01)	0.10 (0.00)	**0.08** (0.00)
	Micro-F1↑	0.74 (0.00)	0.77 (0.03)	0.75 (0.02)	0.80 (0.01)	**0.83** (0.00)
	Macro-F1↑	0.74 (0.00)	0.78 (0.03)	0.74 (0.02)	0.80 (0.01)	**0.83** (0.01)

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* [13]. This method breaks the multi-label task into L binary classification tasks via Binary Decomposition [2] and achieves early classification via fixed halting-point selection [17]. Thus, *LSTM-BD* neither models label relationships nor achieves adaptive early classification.
- *E-LSTM* [4]. We augment this Early Classification method to solve the EMC 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* [11]. 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* [30]. Order-Free Classifier Chains are a recent and powerful approach to multi-label learning when true label orders are unknown (such as the EMC 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 [3]. This approach captures the relationships between labels but requires all timesteps. To make classifications *early*, we use fixed halting points [17], forcing the model's predictions at preset timesteps.

4.3 Implementation Details

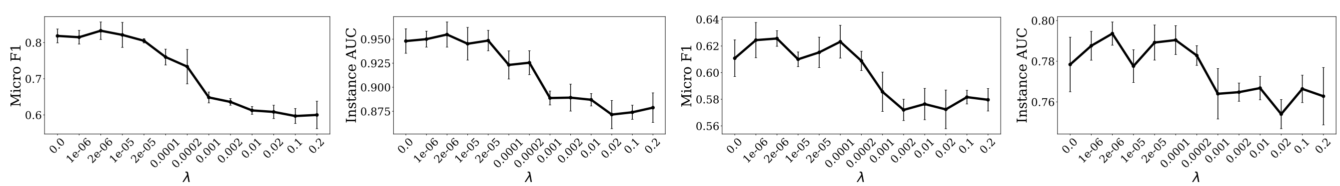
For all datasets, we use an 80% training, 10% validation, and 10% testing split. We use the training set to learn model parameters and the validation set to evaluate the performance of a particular hyperparameter setting (*e.g.*, *nodes-per-layer* or *learning rate*). The testing set is used once to 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 [15] 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`.

4.4 Experimental Results

We evaluate RHC using the `HAR` and `ExtraSensory` datasets described in Section 4.1. We use two groups of metrics: *Instance-AUC*,

Table 3: **Performance (mean (std)) of early multi-label classification on the ExtraSensory dataset. "↓" indicates "the smaller the better" and "↑" indicates "the larger the better".**

Time-Steps Observed	Evaluation Metrics	Methods				
		LSTM-BD [17]	E-LSTM [4]	LSTM-CC [30]	EARLIEST [11]	RHC (ours)
20%	Instance-AUC↑	0.71 (0.00)	0.71 (0.00)	0.74 (0.02)	0.71 (0.00)	**0.78** (0.00)
	Micro-AUC↑	0.70 (0.00)	0.69 (0.00)	**0.72** (0.02)	0.71 (0.00)	0.68 (0.00)
	Macro-AUC↑	0.60 (0.00)	0.62 (0.00)	0.63 (0.01)	0.63 (0.01)	**0.68** (0.00)
	Hamming Loss↓	0.31 (0.00)	0.32 (0.00)	0.31 (0.01)	0.31 (0.00)	**0.27** (0.00)
	Micro-F1↑	0.58 (0.00)	0.55 (0.00)	0.58 (0.02)	0.58 (0.00)	**0.61** (0.00)
	Macro-F1↑	0.48 (0.00)	0.47 (0.00)	0.46 (0.01)	0.47 (0.01)	0.46 (0.00)
40%	Instance-AUC↑	0.74 (0.00)	0.73 (0.00)	0.77 (0.01)	0.74 (0.00)	**0.79** (0.00)
	Micro-AUC↑	0.73 (0.00)	0.71 (0.00)	0.75 (0.01)	0.74 (0.00)	**0.78** (0.00)
	Macro-AUC↑	0.66 (0.00)	0.64 (0.00)	0.68 (0.01)	0.67 (0.00)	**0.70** (0.00)
	Hamming Loss↓	0.32 (0.00)	0.32 (0.00)	0.29 (0.01)	0.27 (0.00)	**0.26** (0.00)
	Micro-F1↑	0.56 (0.00)	0.59 (0.00)	0.60 (0.00)	0.60 (0.00)	**0.62** (0.00)
	Macro-F1↑	0.47 (0.00)	0.52 (0.00)	0.48 (0.02)	**0.53** (0.01)	0.48 (0.00)
60%	Instance-AUC↑	0.76 (0.00)	0.75 (0.01)	0.78 (0.01)	0.77 (0.01)	**0.79** (0.00)
	Micro-AUC↑	0.76 (0.00)	0.73 (0.01)	0.77 (0.01)	0.76 (0.01)	**0.78** (0.00)
	Macro-AUC↑	0.71 (0.00)	0.67 (0.01)	0.70 (0.01)	0.69 (0.01)	0.70 (0.00)
	Hamming Loss↓	0.28 (0.00)	0.32 (0.01)	0.28 (0.01)	0.28 (0.00)	**0.26** (0.00)
	Micro-F1↑	0.61 (0.00)	0.63 (0.01)	0.62 (0.01)	**0.62** (0.01)	**0.62** (0.00)
	Macro-F1↑	0.53 (0.00)	0.56 (0.01)	0.53 (0.01)	**0.55** (0.00)	0.47 (0.00)



(a) Micro-F1 on HAR (b) Instance-AUC on HAR (c) Micro-F1 on ExtraSensory (d) Instance-AUC on ExtraSensory

Figure 4: Observing the effect of λ on Micro-F1 and 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 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 2, RHC consistently achieves the same or better performance on all metrics at each of the three halting points. Most notably, when observing 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 pre-selected halting timesteps.

We observe similar trends on the ExtraSensory dataset, as shown in Table 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.

4.4.1 Parameter Study. RHC has one hyperparameter λ that controls its emphasis on how early predictions *should* be made. We investigate its effect in Figure 4, 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 4a and 4b where the trends for the HAR task are fairly similar to one another. The trends in Figures 4c and 4d also match each other to a significant degree. Overall, however, λ affects datasets differently, demonstrating the need for such hyperparameters.

5 CONCLUSION

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.

6 ACKNOWLEDGEMENTS

This research was supported by the U.S. Dept. of Education grant P200A150306, Worcester Polytechnic Institute through the Arvid Anderson Fellowship, and the National Science Foundation through grants IIS-1718310, IIS-1815866, CNS-1852498, and CNS-1560229. We also thank the Data Science Research group at Worcester Polytechnic Institute for thoughtful feedback and support.

REFERENCES

[1] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. Reyes-Ortiz. 2013. A public domain dataset for human activity recognition using smartphones.. In *ESANN*.
[2] M. Boutell, J. Luo, X. Shen, and C. Brown. 2004. Learning multi-label scene classification. *Pattern Recognition* 37, 9 (2004), 1757 – 1771.
[3] Y.-C. Chen, S.-F.and Chen, C.-K. Yeh, and Y.-C. Wang. 2018. Order-free RNN with visual attention for multi-label classification. In *AAAI*.
[4] D. Dennis, C. Pabbaraju, H. Simhadri, and P. Jain. 2018. Multiple instance learning for efficient sequential data classification on resource-constrained devices. In *NeurIPS*. 10953–10964.
[5] Y. Fujita, N. Kanda, S. Horiguchi, K. Nagamatsu, and S. Watanabe. 2019. End-to-End Neural Speaker Diarization with Permutation-Free Objectives. In *Interspeech*.
[6] M. Ghalwash and Z. Obradovic. 2012. Early classification of multivariate temporal observations by extraction of interpretable shapelets. *BMC Bioinformatics* 13, 1 (2012), 195.
[7] M. Ghalwash, V. Radosavljevic, and Z. Obradovic. 2013. Extraction of interpretable multivariate patterns for early diagnostics. In *ICDM*. 201–210.
[8] M. Ghalwash, V. Radosavljevic, and Z. Obradovic. 2014. Utilizing temporal patterns for estimating uncertainty in interpretable early decision making. In *SIGKDD*. 402–411.
[9] A. Gupta, H. P. Gupta, B. Biswas, and T. Dutta. 2020. An Early Classification Approach for Multivariate Time Series of On-Vehicle Sensors in Transportation. *IEEE Transactions on Intelligent Transportation Systems* (2020).
[10] T. Hartvigsen, C. Sen, S. Brownell, E. Teeple, X. Kong, and E. Rundensteiner. 2018. Early Prediction of MRSA Infections using Electronic Health Records. In *HEALTHINF*. 156–167.
[11] T. Hartvigsen, C. Sen, X. Kong, and E. Rundensteiner. 2019. Adaptive-Halting Policy Network for Early Classification. In *SIGKDD*. 101–110.
[12] G. He, Y. Duan, R. Peng, X. Jing, T. Qian, and L. Wang. 2015. Early classification on multivariate time series. *Neurocomputing* 149 (2015), 777–787.
[13] S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[14] Z. Huang, Z. Ye, S. Li, and R. Pan. 2017. Length Adaptive Recurrent Model for Text Classification. In *CIKM*. 1019–1027.
[15] D. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. In *ICLR*.
[16] Y.-F. Lin, H.-H. Chen, V. Tseng, and J. Pei. 2015. Reliable early classification on multivariate time series with numerical and categorical attributes. In *PAKDD*. 199–211.
[17] S. Ma, L. Sigal, and S. Sclaroff. 2016. Learning activity progression in lstms for activity detection and early detection. In *CVPR*. 1942–1950.
[18] C. Martinez, E. Ramasso, G. Perrin, and M. Rombaut. 2019. Adaptive early classification of temporal sequences using deep reinforcement learning. *Knowledge-Based Systems* (2019).
[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).
[20] U. Mori, A. Mendiburu, S. Dasgupta, and J. Lozano. 2018. Early classification of time series by simultaneously optimizing the accuracy and earliness. *IEEE transactions on neural networks and learning systems* 29, 10 (2018), 4569 – 4578.
[21] U. Mori, A. Mendiburu, E. Keogh, and J. Lozano. 2017. Reliable early classification of time series based on discriminating the classes over time. *Data Mining and Knowledge Discovery* 31, 1 (2017), 233–263.
[22] J. Nam, Y.-B. Kim, E. Mencia, S. Park, R. Sarikaya, and J. Fürnkranz. 2019. Learning Context-dependent Label Permutations for Multi-label Classification. In *ICML*. 4733–4742.
[23] J. Nam, E. Mencía, H. Kim, and J. Fürnkranz. 2017. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *NeurIPS*. 5413–5423.
[24] J. Schulman, N. Heess, T. Weber, and P. Abbeel. 2015. Gradient estimation using stochastic computation graphs. In *NeurIPS*. 3528–3536.
[25] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484.
[26] R. Sutton, D. McAllester, S. Singh, and Y. Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *NeurIPS*. 1057–1063.
[27] C.-P. Tsai and H.-Y. Lee. 2020. Order-free Learning Alleviating Exposure Bias in Multi-label Classification. In *AAAI*.
[28] Y. Vaizman, K. Ellis, and G. Lanckriet. 2017. Recognizing detailed human context in the wild from smartphones and smartwatches. *IEEE Pervasive Computing* 16, 4 (2017), 62–74.
[29] O. Vinyals, S. Bengio, and M. Kudlur. 2017. Order matters: Sequence to sequence for sets. In *ICLR*.
[30] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu. 2016. CNN-RNN: A unified framework for multi-label image classification. In *CVPR*. 2285–2294.
[31] R. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8, 3-4 (1992), 229–256.
[32] Z. Xing, J. Pei, and P. Yu. 2009. Early Prediction on Time Series: A Nearest Neighbor Approach. In *IJCAI*. 1297–1302.
[33] Z. Xing, J. Pei, and P. Yu. 2012. Early classification on time series. *Knowledge and Information Systems* 31, 1 (2012), 105–127.
[34] Z. Xing, J. Pei, P. Yu, and K. Wang. 2011. Extracting interpretable features for early classification on time series. In *SDM*. 247–258.
[35] P. Yang, X. Sun, W. Li, S. Ma, W. Wu, and H. Wang. 2018. SGM: Sequence Generation Model for Multi-label Classification. In *COLING*. 3915–3926.
[36] L. Yao, E. Poblenz, D. Dagunts, B. Covington, D. Bernard, and K. Lyman. 2017. Learning to diagnose from scratch by exploiting dependencies among labels. *arXiv preprint arXiv:1710.10501* (2017).
[37] W. Zhang, D. Jha, E. Laftchiev, and D. Nikovski. 2020. Multi-label Prediction in Time Series Data using Deep Neural Networks. *arXiv preprint* abs/2001.10098 (2020).

APPENDIX

In this appendix we discuss further details to aid in reproducibility of our method and experiments. All code, data, pre-processing, and working experiments are publicly available[1].

6.1 Expanded dataset descriptions

6.1.1 Human Activity Recognition. For the Human Activity Recognition task [1], the six actions performed by participants were *Walking*, *Walking Up Stairs*, *Walking Down Stairs*, *Sitting*, *Standing*, and *Laying*. The distributions (Shown in Figure 5) are relatively balanced across all classes since this is a *scripted* dataset: during collection, each participant performed each action in sequence.

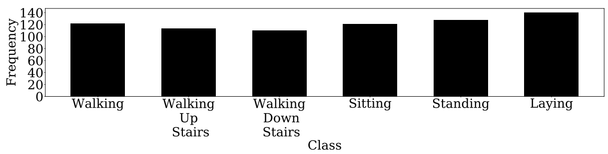


Figure 5: Class label balance in HAR.

6.1.2 ExtraSensory. The *ExtraSensory* dataset [28] was collected "in the wild", where participants were asked to annotate their own actions over a period of time. The researchers running the data collection then collect the annotations into 52 classes. Since some classes are extremely rare (for example "At the bar"), we downsample 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 6. 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 *Sitting* 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 EMC 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.

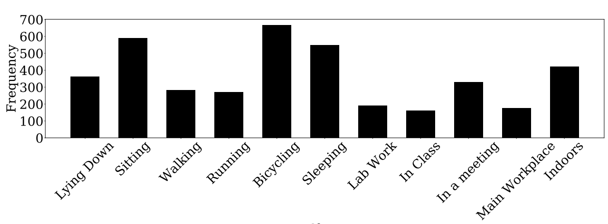


Figure 6: Class label balance in ExtraSensory.

[1] `https://github.com/thartvigsen/RecurrentHaltingChain`

6.2 Implementing the Recurrent Halting Chain

Our models are implemented in PyTorch 1.2 so we present "code" in Figure 7 that roughly emulates PyTorch code. Here we assume that `TransitionModel`, `Disciminator`, `HaltingPolicyNetwork` are pre-defined according to the architectures described in Section 3. In practice during inference, as soon as a class is predicted, it can be returned immediately. As written, this code simply loops until all classes have been halted.

```
def RHC(time_series):
    state = init_zero_vector(V) # V-dimensional state
    y_bar = init_zero_vector(L) # L classes
    final_predictions = init_zero_vector(L)
    for t in range(T): # max of T timesteps
        x = time_series[t] # extract timestep t
        state = TransitionModel(x, state, y_bar)
        y_hat = Discriminator(state)
        halt_probs = HaltingPolicyNetwork(state, y_hat)
        actions = Bernoulli.sample(halt_probs)
        y_bar[(actions == 1) && (y_bar == 0)] = 1
        final_predictions[(actions == 1) && (final_predictions == 0)] = y_hat
        if sum(y_bar) == L: # If all classes are halted, stop reading the time series
            break
    # If a class was never halted, return the final y_hat
    final_predictions[(final_predictions == 0)] = y_hat
    return final_predictions
```

Figure 7: Implementing the Recurrent Halting Chain.