Robust Recurrent Classifier Chains for Multi-Label Learning with Missing Labels

Walter Gerych
Worcester Polytechnic Institute
wgerych@wpi.edu

Thomas Hartvigsen
Massachusetts Institute of Technology
tomh@mit.edu

Luke Buquicchio
Worcester Polytechnic Institute
ljbuquicchio@wpi.edu

Emmanuel Agu
Worcester Polytechnic Institute
emmanuel@wpi.edu

Elke Rundensteiner
Worcester Polytechnic Institute
rundenst@wpi.edu

ABSTRACT

Recurrent Classifier Chains (RCCs) are a leading approach for multi-label classification as they directly model the interdependencies between classes. Unfortunately, existing RCCs assume that every training instance is completely labeled with all its ground truth classes. In practice often only a subset of an instance's labels are annotated, while the annotations for other classes are *missing*. RCCs fail in this missing label scenario, predicting many false negatives and potentially missing important classes. In this work, we propose Robust-RCC, the first strategy for tackling this open problem of RCCs failing for *multi-label missing-label* data. Robust-RCC is a new type of deep recurrent classifier chain empowered to model inter-class relationships essential for predicting the *complete* label set most likely to match the ground truth. The key to Robust-RCC is the design of the Multi Incomplete Label Risk (MILR) function, which we prove to be equal in expectation to the true risk of the ground truth full label set despite being computed from incompletely labeled data. Our experimental study demonstrates that Robust-RCC consistently beats six state-of-of-the-art methods by as much as 30% in predicting the true labels.

CCS CONCEPTS

• **Theory of computation** → **Semi-supervised learning**; • **Computing methodologies** → *Neural networks.*

KEYWORDS

Recurrent classifier chains, Multi-Label Classification, Missing Labels, Incomplete Labels, Deep Learning

ACM Reference Format:
Walter Gerych, Thomas Hartvigsen, Luke Buquicchio, Emmanuel Agu, and Elke Rundensteiner. 2022. Robust Recurrent Classifier Chains for Multi-Label Learning with Missing Labels. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557438

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557438

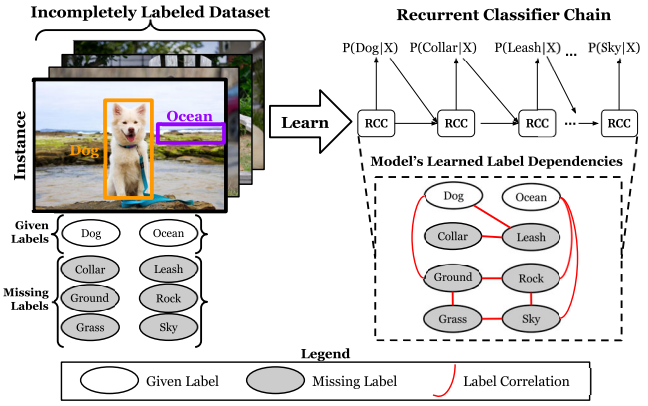

Figure 1: Goal: train a recurrent classifier chain to learn label dependencies on multi-label data with missing labels.

1 INTRODUCTION

Background. Recurrent Classifier Chains (RCCs) are a state-of-the-art approach for multi-label classification [30]. They directly model inter-dependencies between multiple labels, which leads to high classification accuracy [35].

RCCs are often preferred over other multi-label methods because they condition their model of label dependencies on each input instance: the correlation between labels changes based on the attributes of an individual instance [30]. This is superior to many other multi-label methods, such as graph-based methods [9, 47], which typically model *unconditioned*, static label dependencies that do not change based on the input.

Unfortunately, existing RCCs assume that their training data are *fully-labeled*: they perform poorly when training labels are *missing*,[1] meaning when the labels given for an instance are a subset of its *true* labels. This is so, because RCCs wrongly infer that missing labels are always *negative*, i.e., do not apply to the instance. By treating missing labels as negatives during training, RCCs incorrectly predict many false negatives during testing. This results in poor performance on such *multi-label missing-label* (MLML) data.

High false negative rates greatly limit the applicability of RCCs as missing labels are ubiquitous in real-world datasets [37]. For instance, images for training computer vision systems are often

[1] As clearly demonstrated by our experimental study in Section 6.

incompletely labeled by time-pressured and underpaid annotators. Such under-labeling has proven to be more pervasive than many researchers previously thought, cropping up even in popular datasets like ImageNet [13, 37]. Similarly, medical records often have missing labels too: over-worked healthcare staff, often specialists and/or restricted in time or authority, will not assess and thus label each patient for *every* conceivable ailment.

State-of-the-Art. RCCs have gained great popularity in the recent literature [20, 43, 54]. These methods have been applied to many multi-label tasks, including early classification [20], text mining [30], and computer vision [54], while numerous extensions of classifier chains have also been proposed [20, 43, 54]. *However, no RCC model to date has been equipped to handle missing labels.*

Missing labels are pervasive, so learning in their presence is actively researched [44]. While most approaches only study binary classification, some multi-label approaches emerged recently [1, 11, 23, 27, 41, 46, 52]. However, these new approaches make strict assumptions [39, 49, 51, 58]. For instance, it is commonly assumed that a fully-labeled dataset is available alongside an unlabeled or MLML dataset [26, 29, 45], or some explicit negative labels are given (known as *Explicit* MLML) [48, 50, 55]. However, it is unrealistic to expect explicit negative labels in practice. For instance, annotators typically provide labels for what *is* in the given image, and do not give negative labels for every possible thing *not* in the image. The few approaches that do focus on our implicit MLML setting do not directly apply to RCCs, instead requiring other architectures like graph-based models [41] or ranking algorithms [23].

Problem Definition. Our work is the first to extend the leading multi-label classification method, RCCs, to the implicit MLML scenario. More concretely, we address the challenging problem setting where an instance's given set of labels is a subset of its true classes. We do *not* assume that we are given explicit negative labels (we are never told when classes do NOT apply). While *implicit* MLML is the precise name for this setting [28], for ease of readability we henceforth call this setting MLML unless it is contextually unclear.

Our MLML setting matches the most common approach for labeling multi-label data [30]: annotations are assumed to be a set of positive labels [37]. The popular ImageNet benchmark dataset exemplifies MLML, where only positive annotations are given for the images and no explicit negative labels. In the fully-labeled setting, a class being absent from the label set would imply that the instance is a negative instance of that class (the class does not apply to this instance). In contrast, with MLML, the absence of an annotation does not imply that the class is negative. Our goal thus is to train a RCC model that returns the label set that is most likely to match the complete *true* label set rather than some incomplete label set, even when trained only on MLML data without any explicit negative labels.

Technical Challenges. We identify two core challenges in addressing the implicit MLML setting. First, our MLML setting matches the multi-label *Positive Unlabeled* (PU) setting [3] in that we lack explicit negative labels. The lack of negative labels and the ambiguity of unlabeled instances (i.e., being either positive or negative) makes the PU setting classically difficult [3]. Second, it is particularly challenging to learn label dependencies when labels can be missing. For instance, while class 1 and class 2 might be highly correlated, the label for class 2 might be missing from an instance and thus the inferred correlations are weaker. Learning these dependencies is clearly essential for multi-label learning.

Proposed Approach. In this work, we propose Robust-RCC, the first RCC for MLML data. The Robust-RCC learns to model the inter-class dependencies of the *true* labels in order to predict the *complete* label set most likely to match the unobserved *ground truth*.

The Robust-RCC architecture consists of three core components: 1) R-RCC Featurization Network finds a representation of the input data useful for classification.

2) R-RCC Classifier takes in the featurized data and performs *order-free* class predictions, so classes can be predicted in any order. To address the MLML setting, this component is not necessarily punished if it predicts a class not in the annotated label set.

3) R-RCC Prior Estimator estimates the number of positive instances of each class from incompletely labeled data. Knowledge of this quantity empowers us to compute an unbiased loss term that leads the R-RCC Classifier to model the true label dependencies even given MLML data.

More specifically, for the loss function, we introduce a novel *multi-incomplete-label risk* (MILR) function, which allows us to compute an unbiased estimate of the expected multi-label binary cross entropy (BCE) loss between the Robust-RCC's predicted label set and the unobserved *ground truth* label set. This addressed the second challenge of learning label dependencies from MLML data, as this results in the Robust-RCC learning the conditional probability of the true joint label set.

MILR is computed directly from incompletely-labeled instances and class priors, leading to effective training *without negative labels*, while still being an unbiased estimator of the ground truth risk.

Contributions. Our work contributes the following:

- We identify and characterize the important open problem of training RCCs on multi-label missing-label (MLML) data.
- Our proposed method, Robust-RCC, the first solution to this open problem, leverages our novel *multi-incomplete-label risk* (MILR) function to train an accurate RCC from MLML data.
- We theoretically prove that MILR is an unbiased estimate of the risk between the model's prediction and the *true, unobserved* ground truth, yet utilizes only MLML data.
- With a series of rigorous experiments on several real-world datasets, we demonstrate that Robust-RCC outperforms state-of-the-art methods by about 30% on the strictest multi-label metric, Subset Accuracy.

2 RELATED WORK

Classifier Chains. Standard classifier chains (CCs) consist of a sequence of classifiers, each of which is trained to predict a single class while taking in observed data features as well as preceding class labels as input [35]. By conditioning each label prediction on those previously predicted labels, classifier chains succeed to learn joint label dependencies. While these classic methods use independent models for each predicted label [34], most recent works use recurrent neural networks (RNN) [8, 19, 30, 43]. Such *recurrent classifier chains* allow for parameter sharing between label predictions, often leading to better performance [30]. The most recent RCCs

are order-free, meaning, they also learn the best label orderings [8, 43], which we adopt in this work. However, as all RCC methods rely on the strict assumption that all training data are labeled perfectly [8, 30, 43], they fail when trained on MLML data - as we demonstrate in Section 6

Multi-Label Learning with Incomplete Labels. Multi-label classification from incompletely-labeled data is an active area of research [1, 11, 23, 27, 41, 46, 52]. The broad category of learning with incomplete labels encompasses several problem settings [28].

Semi-supervised multi-label learning (SS-ML) assumes that the training data comes in the form of a fully-labeled subset and an unlabeled subset [26, 29, 39, 45, 58]. This does not match our problem setting, where the set of labels *applied to an instance* may be incomplete.

Explicit Multi-Label learning with Missing Labels (Explicit MLML) differ from SS-ML by allowing each *instance* to be partially labeled [28]. They assume that for each class a given instance is given either a positive, negative, or explicitly *missing* label [48–51, 55]. This means that they assume that explicit negative labels are given in addition to some missing labels. As we discussed in the introduction, assuming the availability of negative labels is often unrealistic.

Implicit Multi-Label learning with Missing Labels (Implicit MLML) is our particular problem setting. Implicit MLML methods assume that for each class, an instance is either given a positive label for that class or else receives no label for the class [12, 23, 27, 40, 42, 53, 56]. In other words, no explicit negative labels are given and unlabeled classes could be either positive or negative. Note that implicit MLML methods can be applied to explicit MLML data by simply disregarding the negative labels. Alternatively, explicit MLML methods generally can't be applied to implicit MLML data as these methods may require explicit negative labels.

Existing implicit MLML methods optimize for metrics such as hamming accuracy and ranking loss [23], and are thus not suitable for training RCCs. This is because those metrics can be optimized for without learning the joint conditional interdependencies [10], in contrast to the motivation behind RCCs which is to learn such dependencies. Others require specific architecture choices [58] that make them incompatible with RCCs. *To-date, no method has been proposed that extends RCCs into the MLML setting.*

Positive Unlabeled Learning. Positive Unlabeled (PU) learning is very closely related to implicit MLML. Like MLML, PU learning assumes that some positive labels are given while negative instances are not labeled [3]. Unlike MLML, PU methods are classically *binary* classification problems [16, 38], not multi-label. In this sense, implicit MLML can be seen as being synonymous with multi-label PU. PU learning is also an active area of research [7, 18, 22, 36, 57], with recent works showing that unbiased positive-negative risk minimization can be achieved in both the standard setting [25] and even when the labels are applied with a selection bias [4]. However, due to focusing on binary classification, classic PU methods are not applicable to RCCs. Multi-label PU methods such as RankPU [23] optimize for ranking loss, which can be optimized for without learning label dependencies [10] and is thus not appropriate for training RCCs.

Symbol	Meaning
Lowercase bold symbol	A vector
Uppercase bold symbol	A matrix
$\boldsymbol{y}$	Ground-truth full label vector
$\boldsymbol{y}^*$	Incomplete label vector
$\hat{\boldsymbol{y}}$	Predicted full label set
$[(\cdot)]_k$	kth entry in vector $(\cdot)$
π_k	Class prior of kth class
$\boldsymbol{x}$	Input instance
L	Number of possible classes

Table 1: Notation for commonly used symbols.

3 PROBLEM FORMULATION

Formally, we define our problem as follows: Let $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i^*)\}_{i=1}^n$ be a dataset consisting of n pairs of input features $\boldsymbol{x}_i$ of an instance and its incomplete label sets $\boldsymbol{y}_i^*$. For the sake of readability, we drop the subscript i when referring to a particular instance, when none-ambigious. Let $\boldsymbol{y}^*$ be represented as a vector of labels, and $[\boldsymbol{y}^*]_k$ be the kth element of $\boldsymbol{y}^*$. As $\boldsymbol{y}^*$ is the vector representation of the *incomplete* label set, $[\boldsymbol{y}^*]_k = 1$ implies the kth class applies to the instance, while $[\boldsymbol{y}^*]_k = 0$ implies that the kth class can be either positive or negative—the true value is unknown. This implies that in our incompletely label setting, we have no explicit negative labels for any class. We assume that the labels are missing at random, as is standard [23]. This means that the probability that a true positive instance of a class k is labeled is some constant value c_k; i.e., $p([\boldsymbol{y}^*]_k = 1 \mid [\boldsymbol{y}]_k = 1) = c_k$, where $\boldsymbol{y}$ is the fully labeled ground truth vector such that $[\boldsymbol{y}]_k$ is the true value of the k-th class for this instance. Our goal is to train a classifier $f_\theta : \mathcal{X} \to \mathcal{Y}$, where $\boldsymbol{y} \in \mathcal{Y}$ is the corresponding completely-labeled version of $\boldsymbol{y}^*$, given only observations from $\mathcal{X}$ and corresponding incomplete label vectors $\boldsymbol{y}^*$. Table 1 lists the meaning of the most important notation used in this work.

4 BACKGROUND: RCCS

RCCs model the conditional joint probability of the labels [30]. More formally, they model $p(\boldsymbol{y}|\boldsymbol{x}) = p([\boldsymbol{y}]_1, [\boldsymbol{y}]_2, ..., [\boldsymbol{y}]_L|\boldsymbol{x})$. They accomplish this by factorizing the joint probability as follows:

$$p([\boldsymbol{y}]_1, [\boldsymbol{y}]_2, ..., [\boldsymbol{y}]_L|\boldsymbol{x}) = p([\boldsymbol{y}]_1|\boldsymbol{x}) \prod_{i=2}^{L} p([\boldsymbol{y}]_i|[\boldsymbol{y}]_{<i}, \boldsymbol{x}), \quad (1)$$

where $[\boldsymbol{y}]_{<i} = ([\boldsymbol{y}]_1, [\boldsymbol{y}]_2, ..., [\boldsymbol{y}]_{i-1})$. RCCs model the above as a recurrent neural network. The recurrent network reads in the feature attributes along with the observations of each class sequentially; i.e., at the ith step it reads in the observation for the ith class. It thus parameterizes $[\boldsymbol{y}]_{<i}$ as h_{i-1}, where h_{i-1} is the hidden state of the recurrent network at the $i-1th$ step. Likewise, it gives $p([\boldsymbol{y}]_i|[\boldsymbol{y}]_{<i}, \boldsymbol{x})$ as the output of a feed forward network conditioned on h_{i-1}.

As described, the RCC factorizes the classes in a predefined order; i.e., class 2 is predicted after class 1. However, recent RCC methods [8, 43] can predict the classes in an arbitrary order that differs instance-to-instance. Thus, they instead provide an alternative

factorize to Equation 2 as:

$$p([\boldsymbol{y}]_1, [\boldsymbol{y}]_2, ..., [\boldsymbol{y}]_L|\boldsymbol{x}) = \prod_{i \in \mathcal{O}(\boldsymbol{x})}^{L} p([\boldsymbol{y}]_i|[\boldsymbol{y}]_{<i}, \boldsymbol{x}), \tag{2}$$

where $\mathcal{O}(\boldsymbol{x})$ is an ordered list of class indices specific to instance $\boldsymbol{x}$.

5 METHODOLOGY

5.1 Overview of Robust-RCC

In this work, we propose the Robust-RCC, the first RCC for MLML data. The Robust-RCC learns the true conditional label dependencies from incompletely labeled data. Robust-RCC is composed of a featurization network and a recurrent network with the later optimized using a novel reformulation of multi-label risk which we derive in this work.

First, the R-RCC Featurization Network transforms an input instance to a latent vector representation. Second, the latent representation is then fed to a novel *R-RCC Classifier* (R-RCC Backbone), which learns the conditional distribution of the *ground truth* label vector given MLML training data. We achieve this training the R-RCC Backbone using a novel multi-incomplete-label risk function (MILR), which reformulates the multi-label risk to be computable from incomplete labels. This is achieved using knowledge of the class priors, estimated by the R-RCC Prior Estimator from incompletely labeled data.

We first describe the novel risk function. Then, we describe the architecture of Robust-RCC in detail and show how the R-RCC Backbone can learn the order in which to predict classes even when given incomplete labels.

5.2 Reformulating the Multi-Label Risk.

In the traditional fully labeled setting, RCCs are trained by minimizing the Binary Cross Entropy (BCE) between the predicted label sets and the true label sets [30]. In other words, they aim to find the parameters θ^* of a model $f()$ that minimizes:

$$\theta^* = \arg\min_{\theta} \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y} \sim P_{X,Y}} \text{BCE}(f_\theta(\boldsymbol{x}), \boldsymbol{y}), \tag{3}$$

where $P_{X,Y}$ is the joint probability of features and ground truth label vectors. However, we observe that Equation 3 cannot be calculated in the MLML setting due to requiring the expectation over the completely-labeled instances $\boldsymbol{y}$, to which we have no access. Instead, we propose the following multi-incomplete-label risk that can be computed given only implicit MLML data while still being minimized by the ground-truth label vector:

$$\begin{aligned} MILR = \sum_{k}^{L} \Big[& \pi_k \mathop{\mathbb{E}}_{[\boldsymbol{y}^*]_k^+} L^+(f_\theta(\boldsymbol{x})[k]) \\ & + (\pi_k c_k - \pi_k) \mathop{\mathbb{E}}_{[\boldsymbol{y}^*]_k^+} L^-([f_\theta(\boldsymbol{x})]_k) \\ & + (1 - \pi_k c_k) \mathop{\mathbb{E}}_{[\boldsymbol{y}^*]_k^-} L^-([f_\theta(\boldsymbol{x})]_k) \Big]. \end{aligned} \tag{4}$$

Here, π_k refers to the k-th class prior, $\pi_k = p([\boldsymbol{y}]_k = 1)$, $c_k = \frac{p([\boldsymbol{y}^*]_k=1)}{\pi_k}$, and $\mathbb{E}_{[\boldsymbol{y}^*]_k^{+/-}}$ refers to $\mathbb{E}_{\boldsymbol{x},[\boldsymbol{y}^*]_k \sim (X,[Y^*]_k=1/0)}$. L^- and L^+ refer to the components of the *decomposed* BCE loss, such that $L^{+/-}$ for a given class is the loss incurred by BCE for that class assuming that the ground truth is positive or negative, respectively. While Positive Unlabeled risk functions have been proposed in the setting of *binary* classification [14], this is the first general PU risk formulation for multi-label learning.

Equation 4 succeeds to optimize the multi-label BCE, requiring only expectations over positive instances ($\mathbb{E}_{[\boldsymbol{y}^*]_k^+}$) and unlabeled instances ($\mathbb{E}_{[\boldsymbol{y}^*]_k^-}$) per individual class. It is important to note that we can approximate these expectations from MLML data, while expectations over negative instances would not be. However, Equation 4 is not useful if it were to produce biased label sets. Fortunately, we can establish that this is not the case, as stated in the theorem below.

Theorem 1. *The expected value of the MILR risk function computed from incompletely labeled data is equal in expectation to the expectation of multi-label binary cross entropy loss computed from the ground-truth full label vectors.*

Proof. Let $f_\theta(x)$ be the estimate of $\boldsymbol{y}$ for instance x such that f_θ is the output of a probabilistic RCC parameterized by θ and $[f_\theta(x)]_k$ is the RCC's estimate of $p([\boldsymbol{y}]_k|x)$. To train a standard RCC, we would minimize the BCE:

$$\theta^* = \arg\min_{\theta} \mathop{\mathbb{E}}_{x,\boldsymbol{y} \in (X,Y)} \text{BCE}(f_\theta(x), \boldsymbol{y}), \tag{5}$$

where $\text{BCE}(f_\theta(x), \boldsymbol{y})$ is defined as

$$\begin{aligned} \text{BCE}(f_\theta(x), \boldsymbol{y}) = & \frac{1}{L} \sum_{k}^{L} -[\boldsymbol{y}]_k log([f_\theta(x)]_k) \\ & - (1 - [\boldsymbol{y}]_k) log(1 - [f_\theta(x)]_k). \end{aligned} \tag{6}$$

Inspired by binary Positive Unlabeled (PU) methods [14], we reformulate the risk in Equation 6 to be expressed in terms of only positive and unlabeled instances *in the multi-label setting*.

Let $L^+([f_\theta(x)]_k) = \log([f_\theta(x)]_k)$ and $L^-(]f_\theta(x)]_k) = \log(1 - [f_\theta(x)]_k)$. Then, because the expectation is a linear operator, Equation 5 can be rewritten as:

$$\begin{aligned} \theta^* = \arg\min_{\theta} \sum_{k}^{L} \mathop{\mathbb{E}}_{x,[\boldsymbol{y}]_k \in (X,[Y]_k)} \Big[& L^+([f_\theta(x)]_k) \\ & + L^-([f_\theta(x)]_k) \Big]. \end{aligned} \tag{7}$$

Let $\mathbb{E}_{[\boldsymbol{y}]_k^{+/-}}$ refer to $\mathbb{E}_{\boldsymbol{x},[\boldsymbol{y}]_k \sim (X,[Y]_k=1/0)}$. Then, we split the expectation of the BCE into an expectation of positive and negative instances for each class:

$$\begin{aligned} \theta^* = \arg\min_{\theta} \sum_{k}^{L} \Big[& \pi_k \mathop{\mathbb{E}}_{[Y]_k^+} L^+([f_\theta(x)]_k) \\ & + (1 - \pi_k) \mathop{\mathbb{E}}_{[Y]_k^-} L^-([f_\theta(x)]_k) \Big], \end{aligned} \tag{8}$$

where π_k refers to the k-th class prior, $\pi_k = p([\boldsymbol{y}]_k = 1)$. Next, note that the expectation of L^- over all instances of a given class can be rewritten as:

$$\begin{aligned} \mathop{\mathbb{E}}_{[Y]_k} L^-([f_\theta(x)]_k) = & \pi_k \mathop{\mathbb{E}}_{[Y]_k^+} L^-([f_\theta(x)]_k) \\ & + (1 - \pi_k) \mathop{\mathbb{E}}_{[Y]_k^-} L^-([f_\theta(x)]_k) \end{aligned} \tag{9}$$

For each class, let $p(x)$ be the *PDF* of the features, and $p_{+/-/\ell/u}(x)$ be the PDF of the positive instance, negative instances, labeled positive instances, and unlabeled instances respectively. Then, for each class k, $x \sim \pi_k p_+(x) + (1-\pi_k)p_-(x) \sim \pi_k c_k p_\ell(x) + (1-\pi_k c_k)p_u(x)$ [3], where $c_k = \frac{p([\boldsymbol{y}^*]_k=1)}{\pi_k}$ (which can be explicitly calculated from the data given π_k). With that in mind, we can rewrite the expectation of L^- over the negative instances in terms of the expectation over all instances and the expectation over positive instances:

$$\begin{aligned}(1-\pi_k)\mathop{\mathbb{E}}_{[Y]_k^-} L^-([f_\theta(x)]_k) = -\pi_k \mathop{\mathbb{E}}_{[Y]_k^+} L^-([f_\theta(x)]_k) \\ + \mathop{\mathbb{E}}_{[Y]_k} L^-([f_\theta(x)]_k).\end{aligned} \tag{10}$$

And likewise the unconditioned expectation over all instances can be written in terms of positive and unlabeled distributions as

$$\begin{aligned}\mathop{\mathbb{E}}_{[Y]_k} L^-([f_\theta(x)]_k) =& \pi_k c_k \mathop{\mathbb{E}}_{[Y]_k^+} L^-([f_\theta(x)]_k) \\ &+(1-\pi_k c_k)\mathop{\mathbb{E}}_{[Y]_k^{*-}} L^-([f_\theta(x)]_k),\end{aligned} \tag{11}$$

where $\mathbb{E}_{[Y]_k^{*-}} = \mathbb{E}_{x,[\boldsymbol{y}]_k \in (X,[Y^*]_k=0)}$.

We can replace the expectation over negative instances in Equation 8 with the right hand side of Equation 10 and the expectation over unconditioned instances with Equation 11 to arrive at our reformulated BCE loss function:

$$\begin{aligned}\theta^* = \arg\min_\theta \sum_k^L \Big[&\pi_k \mathop{\mathbb{E}}_{[Y]_k^+} L^+([f_\theta(x)]_k) + \\ &(\pi_k c_k - \pi_k)\mathop{\mathbb{E}}_{[Y]_k^+} L^-([f_\theta(x)]_k) \\ &+(1-\pi_k c_k)\mathop{\mathbb{E}}_{[Y]_k^{*-}} L^-([f_\theta(x)]_k)\Big].\end{aligned} \tag{12}$$

Lastly, as we assume that there is no selection bias in which classes are labeled for each instance, we can write the above expectation over positive instances of the true label set with positively labeled instances of the incomplete label set:

$$\begin{aligned}\theta^* = \arg\min_\theta \sum_k^L \Big[&\pi_k \mathop{\mathbb{E}}_{[Y^*]_k^+} L^+([f_\theta(x)]_k) + \\ &(\pi_k c_k - \pi_k)\mathop{\mathbb{E}}_{[Y^*]_k^+} L^-([f_\theta(x)]_k) \\ &+(1-\pi_k c_k)\mathop{\mathbb{E}}_{[Y^*]_k^-} L^-([f_\theta(x)]_k)\Big].\end{aligned} \tag{13}$$

□

Theorem 1 implies that we can replace the BCE risk with the MILR risk (Equation 4) in order to train RCCs, without introducing bias into our predicted label sets.

Equation 4 requires us to compute the expectations of losses over feature-label pairs. During the training, since we have finite training data, we thus can replace Equation 4 with the *empirical MILR*, $\overline{MILR}$:

$$\begin{aligned}\overline{MILR}(f_\theta) = \sum_k^L \Big[&\frac{\pi_k}{n} \sum_{\boldsymbol{x}_i:[\boldsymbol{y}_i^*]_k=1}^{n} L^+([f_\theta(\boldsymbol{x}_i)]_k) \\ &+\frac{(\pi_k c_k - \pi_k)}{n'} \sum_{\boldsymbol{x}_j:[\boldsymbol{y}_j^*]_k=1}^{n'} L^-([f_\theta(\boldsymbol{x}_j)]_k) \\ &+\frac{(1-\pi_k c_k)}{n''} \sum_{\boldsymbol{x}_m:[\boldsymbol{y}_m^*]_k=0}^{n''} L^-([f_\theta(\boldsymbol{x}_m)]_k)\Big]\end{aligned} \tag{14}$$

Through Equation 14, we succeed to present the very first risk function that can be used to train an RCC to model the true conditional class distribution even for MLML data. Most importantly, Equation 14 does not require negative labels, yet it minimizes Equation 3.

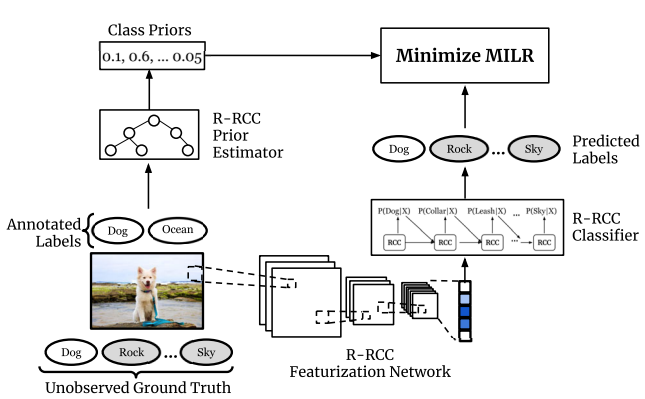

Figure 2: Architecture of Robust-RCC.

5.3 Robust-RCC Architecture.

We next discuss the Robust-RCC's architecture. There are three main components of the Robust-RCC's architecture: the R-RCC Featurization Network, the R-RCC Backbone model, and the R-RCC Prior Estimator. These components are trained together to minimize Equation 14.

First, the R-RCC Featurization Network, $\mathcal{F}$, produces a latent vector representations of input instances x, namely, $\boldsymbol{v}' = \mathcal{F}(\boldsymbol{x})$. In this work, we use ResNet-18 [21] pre-trained on ImageNet to produce a featurized representation, as we focus on image datasets. However, many alternate options could equally be plugged in for this component based on the nature of the task at hand (i.e., a transformer for text data).

Second, the R-RCC Backbone is a recurrent network that takes in this latent representation and produces one class probability per step. At step t, the R-RCC Backbone outputs $[f_\theta(\boldsymbol{x})]_{c_t}$, such that $[f_\theta(\boldsymbol{x})]_{c_t} = p([\boldsymbol{y}]_{c_t} = 1|\boldsymbol{x})$ where c_t is the class predicted at step t. At each step, the input is $\boldsymbol{v} = \boldsymbol{v}' \oplus [f_\theta(\boldsymbol{x})]_{c_{t-1}}$, the concatenation of the feature representation $\boldsymbol{v}'$ with the previous class probability.

We use a *gated recurrent unit* (GRU) for the R-RCC Backbone, though in practice any recurrent network could be used. Thus the

predicted class probabilities $[f_\theta(\boldsymbol{x})]_{c_t}$ are given as:

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r \boldsymbol{v} + \boldsymbol{U}_r \boldsymbol{h}_{t-1} + \boldsymbol{b}_r) \tag{15}$$

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_z \boldsymbol{v} + \boldsymbol{U}_z \boldsymbol{h}_{t-1} + \boldsymbol{b}_z) \tag{16}$$

$$\boldsymbol{s}_t = \Phi(\boldsymbol{W}_a \boldsymbol{v} + \boldsymbol{U}_a(\boldsymbol{r}_t \odot \boldsymbol{a}_t - 1) + \boldsymbol{b}_a) \tag{17}$$

$$\boldsymbol{a}_t = \boldsymbol{z}_t \odot \boldsymbol{a}_{t-1} + (1 - \boldsymbol{z}_t) \odot \boldsymbol{s}_t \tag{18}$$

$$[f_\theta(\boldsymbol{x})]_{c_t} = \sigma(\boldsymbol{W}_f \boldsymbol{a}_t + \boldsymbol{b}_f), \tag{19}$$

where $\boldsymbol{W}_{r,z,a}$ and $\boldsymbol{U}_{r,z,a}$ are the weight matrices of the GRU and $\boldsymbol{W}_f \in \mathbb{R}^{|h_t| \times 1}$ is the weight matrix of a feed-forward layer used to convert the hidden representation of the GRU into a class probability. σ is the sigmoid function and Φ is the hyperbolic tangent function.

The final component of the Robust-RCC is the R-RCC Prior Estimator, which estimates the frequency of each class: $\pi_k = p([\boldsymbol{y}]_k = 1)$ for $k = 1, ..., L$. This value is required by our risk function, as discussed in Section 5.2. If the class priors are known, they may be substituted here. Otherwise, we use TiCE [2], a leading prior estimation method, to estimate the class prior of each class. TiCE utilizes top-down decision tree induction to estimate the labeling frequency $c = p([\boldsymbol{y}^*]_k = 1 | [\boldsymbol{y}]_k = 1)$ in subdomains of the data. Under the assumption that there is no bias in the labeling, subdomains with a higher ratio of labeled to unlabeled instances provide a better estimate of the labeling frequency. The class prior can be recovered form the labeling frequency by the simple conversion $\pi_k = p([\boldsymbol{y}^*]_k = 1)/c$ [3].

Order-free Classification. The order in which classes are predicted should be learned and not pre-determined, because $p(\boldsymbol{y}|\boldsymbol{x})$ can be factorized into any order of conditional probabilities. However, most RCCs are forced to predict labels in a pre-defined order (often frequent-to-rare or rare-to-frequent [30, 35]) despite the large impact of such selection on performance [35]. To overcome this, recent work has shown that classification can be improved with *order-free* approaches, where the RCC learns the order in which to predict classes based on the input. In our work, we use such an order-free approach inspired by [8]. We show experimentally that this choice is well justified. To predict a new class label at step t, our model chooses from the set of previously not yet predicted labels $\mathbb{C}'_t$:

$$c_t = \arg\max_{c' \in \mathbb{C}'_t} [p_t]_{c'} \tag{20}$$

$$f_{\theta_t} = f_{\theta_{t-1}} + [p_t]_{c_t} \tag{21}$$

$$\mathbb{C}'_t = \mathbb{C}'_{t-1} - \{c_t\}, \tag{22}$$

where c_t is the prediction of the class predicted at the t-th step, p_t is the predicted distribution over labels at step t, and $[p_t]_{c_t}$ is the marginal for that class.

Algorithm 1: Training algorithm for Robust-RCC

```
function train_Robust-RCC(D, H, f_θ, prior_identifier);
Input  : Training dataset D, featurization network H,
         order-free RCC classifier f_θ, class prior
         identification method prior_identifier
Output : Parameters θ*
for k in classes do
    π_k = prior_identifier(X, Y[k])
end
for epoch ← 1 to num_epochs do
    for batch in batches do
        for x_b, y*_b in batch do
            for c ← 1 to L do
                V_{b,c} = [H(x_b), f_θ(V_{b,c-1})]
                predict f_θ(V_b) using Equation 19 and 22
            end
        end
        Calculate loss according to Equation 14
        Update θ according gradient descent
    end
end
return θ
```

6 EXPERIMENTS

6.1 Datasets and Metrics.

We evaluate the Robust-RCC on the following three multi-label datasets using four metrics.

PASCAL VOC 2007[2] [17]: This standard multi-label image dataset consists of 9,963 natural images.

Scene[3] [5]: This dataset contains 2407 scenery images, each with up to six labels: *beach*, *sunset*, *fall foliage*, *field*, *mountain* and *urban*. Instead of using ResNet-18, these images have already been featurized into 294-dimensional vectors corresponding to the spatial color moments in the LUV space.

Corel 5k[4] [15]: This dataset is made up of 5,000 images taken from the Corel Photo Gallery.

These three datasets were chosen as they are standard image datasets that are naturally multi-label.

Feature Representations Each method used a feature representation of the images in each dataset. For the Corel 5k[15] and the PASCAL VOC 2007[17] datasets, we used a pretrained ResNet-18 [21] model to featurize the input images into 512-dimensional vectors. Specifically, we used the pretrained ResNet-18 model available in PyTorch, and extracted the feature representations from the final average pooling layer. The Scene[5] dataset was already featurized into 294-dimensional vectors corresponding to the spatial color moments in the LUV space, so we did not use the ResNet-18 model on this dataset and instead used these pre-computed features.

We use **four standard multi-label metrics**: *subset accuracy*, *hamming loss*, *macro F1*, and *micro F1*. The subset accuracy is of particular interest to us, as optimizing for this metric means that the model must learn the dependencies between labels [10]. We report on the top 10 labels for each dataset.

6.2 Compared Methods.

We compare Robust-RCC against the following state-of-the-art methods for learning with incomplete labels:

[2] http://host.robots.ox.ac.uk/pascal/VOC/voc2007/, https://www.flickr.com/help/terms

[3] http://www.uco.es/kdis/mllresources/#SceneDesc, license: PDDL

[4] https://github.com/corel-5k-pytorch/corel-5k, license: Non-comercial use only

Metric	Percent Labeled	Approaches						
		R-RCC (ours)	PU CC	SMiLE	RankPU	CleanLab RCC	CleanLab CC	RCC
Subset Accuracy	10%	**0.395±0.012**	0.000±0.000	0.068±0.022	0.089±0.082	0.000±0.000	0.000±0.000	0.000±0.000
	20%	**0.557±0.018**	0.021±0.017	0.260±0.015	0.073±0.049	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.563±0.056**	0.154±0.035	0.347±0.012	0.041±0.026	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.599±0.036**	0.323±0.028	0.419±0.010	0.162±0.053	0.000±0.000	0.000±0.000	0.021±0.000
	50%	**0.575±0.028**	0.465±0.036	0.468±0.008	0.103±0.073	0.010±0.008	0.007±0.010	0.170±0.000
Hamming Loss	10%	**0.084±0.003**	0.127±0.000	0.118±0.003	0.181±0.029	0.127±0.000	0.127±0.000	0.127±0.000
	20%	**0.061±0.002**	0.124±0.002	0.093±0.003	0.199±0.043	0.127±0.000	0.127±0.000	0.127±0.000
	30%	**0.059±0.006**	0.106±0.005	0.081±0.001	0.207±0.033	0.127±0.000	0.127±0.000	0.127±0.000
	40%	**0.054±0.003**	0.083±0.005	0.071±0.001	0.152±0.030	0.127±0.000	0.127±0.000	0.124±0.006
	50%	**0.058±0.002**	0.065±0.004	0.065±0.001	0.177±0.028	0.125±0.001	0.126±0.001	0.106±0.010
Macro F1	10%	0.401±0.024	0.000±0.000	0.034±0.008	**0.589±0.050**	0.000±0.000	0.000±0.000	0.000±0.000
	20%	**0.646±0.029**	0.013±0.013	0.169±0.013	0.587±0.074	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.670±0.046**	0.087±0.025	0.283±0.013	0.585±0.061	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.708±0.026**	0.274±0.047	0.374±0.019	0.638±0.053	0.000±0.000	0.000±0.000	0.034±0.067
	50%	**0.678±0.018**	0.489±0.046	0.446±0.015	0.619±0.048	0.006±0.006	0.006±0.006	0.243±0.161
Micro F1	10%	**0.626±0.016**	0.000±0.000	0.122±0.045	0.551±0.042	0.000±0.000	0.000±0.000	0.000±0.000
	20%	**0.749±0.011**	0.038±0.031	0.427±0.030	0.545±0.049	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.765±0.019**	0.284±0.056	0.535±0.013	0.536±0.035	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.786±0.010**	0.511±0.041	0.617±0.010	0.607±0.044	0.000±0.000	0.000±0.000	0.041±0.094
	50%	**0.771±0.008**	0.659±0.032	0.665±0.007	0.576±0.039	0.023±0.010	0.025±0.010	0.293±0.142

Table 2: Performance of each method on the `Pascal VOC 2007` benchmark dataset.

Positive Unlabeled Classifier Chains (PU CC) [16, 34]. We train a classifier chain using the standard PU method modification technique [16], as proposed in [42].

SMiLE [41]: This recently-proposed method for learning from incomplete labels uses a graphical model to learn correlations between classes. It optimizes its predictions to preserve the learned class correlations. Unlike our method, *SMiLE* learns the *unconditional* class relations rather than the *conditional* relations.

RankPU [23]: Similar to Robust-RCC, *RankPU* extends Positive Unlabeled learning to the multi-label setting. However, *RankPU* is designed to optimize for ranking algorithms and is not applicable to RCCs.

CleanLab [32]: This identifies instances that are likely mislabeled and removes them before training a classifier. As *CleanLab* is designed to work with any probabilistic classifier, we compare against two versions: one using an RCC as classifier, and the other using the ensemble-based classifier chain.

Recurrent Classifier Chain [30]: We compare against an RCC that treats all unlabeled instances as true negatives. This is the standard approach for maximizing subset accuracy in the fully-labeled setting. We expect other methods to outperform this approach as it does not naturally account for the incompletely labeled nature of the data.

6.3 Implementation Details.

Base RCC Architecture. Robust-RCC, RankPU[23], the Positive Unlabeled Classifier Chain (PUCC) [16, 34], and the Recurrent Classifier Chain (RCC) [30] were each implemented in PyTorch [33]. The recurrent methods (Robust-RCC and RCC) consisted of a 1-layer GRU with a hidden space size of 100. Additionally, each recurrent method had a 1-layer feed forward network to map from the 100 dimensional latent space into prediction probabilities.

Base Feed Forward Network Architecture. The non-recurrent methods (PUCC and RankPU) consisted of a feed-forward network that mapped from the feature space to a 100 dimensional latent space, replacing the GRU of the recurrent methods. These methods likewise had an additional feed-forward layer to map from the latent space into prediction probabilities.

CleanLab Implementation. The CleanLab methods (CleanLab CC and CleanLab RCC) [32] used the above feed-forward and GRU models respectively. We used the publicly available code for CleanLab in order to identify and remove the unlabeled positives prior to training the classifier components.

SMiLE Implementation. We did not implement our own version of SMiLE [41], as the authors had made the code for this method publicly available. We used their code[5] and the parameter settings used in their paper, although we modified the neighbor parameter to 400. This is higher than the number used in their paper, and was modified as the default value produced 0 subset accuracy for nearly all runs. We found the value of 400 for the neighbor hyperparameter to produce optimal results for this method.

Training Hyperparameters. For each method, we used a batch size of 128 and a learning rate of 0.001. We used the Adam optimizer [24] and PyTorch's exponential learning rate scheduler with gamma set to 0.99. Each method was trained until convergence for 200 epochs.

[5] https://github.com/Jopepato/SMiLE

Metric	Percent Labeled	Approaches						
		R-RCC (ours)	PU CC	SMiLE	RankPU	CleanLab RCC	CleanLab CC	RCC
Subset Accuracy	10%	**0.308**±0.095	0.000±0.000	0.060±0.059	0.113±0.054	0.000±0.000	0.000±0.000	0.000±0.000
	20%	**0.424**±0.116	0.000±0.000	0.068±0.062	0.181±0.125	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.474**±0.115	0.001±0.001	0.109±0.071	0.148±0.074	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.509**±0.087	0.010±0.005	0.078±0.066	0.123±0.027	0.000±0.000	0.000±0.000	0.048±0.078
	50%	**0.415** ±0.039	0.021±0.008	0.099±0.056	0.142±0.024	0.134±0.072	0.010±0.004	0.230±0.062
Hamming Loss	10%	**0.149**±0.016	0.181±0.000	0.347±0.124	0.264±0.035	0.181±0.000	0.181±0.000	0.181±0.000
	20%	**0.140**±0.008	0.181±0.000	0.347±0.083	0.226±0.057	0.181±0.000	0.181±0.000	0.181±0.000
	30%	**0.124**±0.016	0.181±0.000	0.305±0.111	0.235±0.033	0.181±0.000	0.181±0.000	0.181±0.000
	40%	**0.128**±0.013	0.180±0.002	0.374±0.115	0.249±0.023	0.181±0.000	0.181±0.000	0.173±0.013
	50%	**0.127**±0.008	0.178±0.001	0.314±0.081	0.233±0.027	0.158±0.022	0.179±0.001	0.143±0.012
Macro F1	10%	0.305±0.061	0.000±0.000	0.261±0.030	**0.575**±0.028	0.000±0.000	0.000±0.000	0.000±0.000
	20%	0.605±0.056	0.000±0.000	0.274±0.016	**0.606**±0.046	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.646**±0.057	0.003±0.003	0.310±0.035	0.606±0.027	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.698**±0.017	0.009±0.017	0.308±0.029	0.600±0.020	0.000±0.000	0.000±0.000	0.079±0.128
	50%	**0.683**±0.025	0.037±0.012	0.322±0.029	0.611±0.032	0.233±0.100	0.174±0.060	0.347±0.084
Micro F1	10%	0.344±0.072	0.000±0.000	0.267±0.023	**0.538**±0.040	0.000±0.000	0.000±0.000	0.000±0.000
	20%	**0.635**±0.033	0.000±0.000	0.278±0.015	0.585±0.053	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.667**±0.027	0.002±0.003	0.307±0.037	0.581±0.030	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.702**±0.009	0.009±0.017	0.305±0.024	0.573±0.021	0.000±0.000	0.000±0.000	0.086±0.141
	50%	**0.692**±0.011	0.038±0.014	0.318±0.028	0.585±0.023	0.241±0.020	0.175±0.042	0.371±0.087

Table 3: Performance of each method on the Scene benchmark dataset.

Metric	Percent Labeled	Approaches						
		R-RCC (ours)	PU CC	SMiLE	RankPU	CleanLab RCC	CleanLab CC	RCC
Subset Accuracy	10%	**0.139±0.042**	0.000±0.000	0.005±0.005	0.021±0.018	0.000±0.000	0.000±0.000	0.000±0.000
	20%	**0.247±0.023**	0.001±0.002	0.032±0.015	0.049±0.037	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.293±0.046**	0.020±0.019	0.077±0.012	0.045±0.013	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.321±0.019**	0.077±0.022	0.098±0.010	0.031±0.032	0.000±0.000	0.000±0.000	0.000±0.000
	50%	**0.304±0.048**	0.158±0.036	0.133±0.016	0.031±0.037	0.003±0.004	0.006±0.011	0.014±0.022
Hamming Loss	10%	**0.137±0.008**	0.156± 0.000	0.155±0.000	0.259±0.028	0.156±0.000	0.156±0.000	0.156±0.000
	20%	**0.122±0.005**	0.156± 0.000	0.152±0.001	0.237±0.044	0.156±0.000	0.156±0.000	0.156±0.000
	30%	**0.119±0.009**	0.152± 0.002	0.148±0.001	0.246±0.016	0.156±0.000	0.156±0.000	0.156±0.000
	40%	**0.118±0.005**	0.140± 0.003	0.143±0.001	0.270±0.069	0.156±0.000	0.156±0.000	0.156±0.000
	50%	**0.120±0.009**	0.124± 0.006	0.138±0.002	0.243±0.037	0.155±0.001	0.154±0.003	0.153±0.003
Macro F1	10%	0.313±0.075	0.000±0.000	0.009±0.011	**0.508**±0.033	0.000±0.000	0.000±0.000	0.000±0.000
	20%	0.0493±**0.040**	0.004±0.008	0.051±0.019	**0.521**±0.038	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.542±0.044**	0.035±0.016	0.109±0.017	0.520±0.029	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.582±0.046**	0.136±0.020	0.149±0.014	0.512±0.043	0.000±0.000	0.000±0.000	0.000±0.000
	50%	**0.585±0.012**	0.274±0.040	0.199±0.015	0.536±0.044	0.030±0.016	0.035± 0.020	0.068±0.063
Micro F1	10%	0.331±.093	0.000±0.000	0.006±0.006	**0.494**±0.021	0.000±0.000	0.000±0.000	0.000±0.000
	20%	0.493±0.030	0.002±0.004	0.044±0.016	**0.526**±0.049	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.567±0.036**	0.048±0.033	0.107±0.014	0.521±0.015	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.596±0.037**	0.187±0.035	0.155±0.015	0.505±0.049	0.000±0.000	0.000±0.000	0.000±0.000
	50%	**0.607±0.008**	0.350±0.059	0.217±0.020	0.523±0.037	0.016±0.007	0.021± 0.020	0.051±0.050

Table 4: Performance of each method on the Corel 5k benchmark dataset.

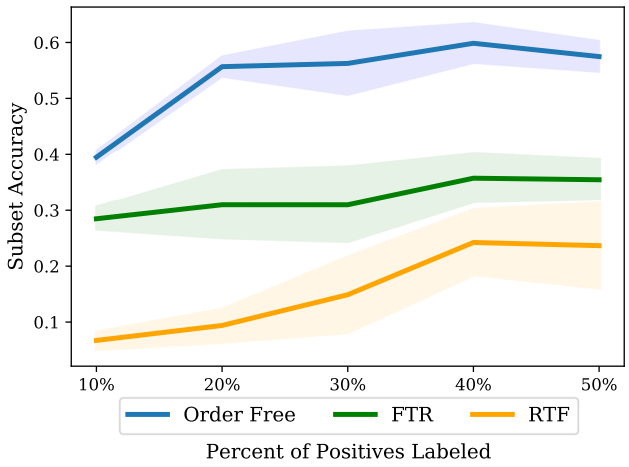

Figure 3: Order-free compared to FTR and RTF Robust-RCC on the PASCAL VOC dataset. The shaded region is the 95% confidence interval.

Experiments were performed on a computing cluster, using a Intel(R) Xeon(R) Platinum 263 8160 CPU @ 2.10GHz CPU, an NVIDIA Tesla V100 SXM2 GPU, and 128 GB of RAM.

6.4 Classification with Incompletely Labeled Data.

We first demonstrate that the Robust-RCC classifies implicit MLML data more accurately than the five state-of-the-art alternatives. To do this, inspired by the approach taken by many other incomplete labeling experiments [2, 6, 14, 16, 23, 41], we remove various amounts of labels such that for each dataset only 10% to 50% of the positive instances are labeled. Positive instances from each class thus had the same labeling probability. The results for the PASCAL VOC, Scene, and Corel 5k datasets are shown in Tables 2, 3, and 4, respectively. Notably, the Robust-RCC routinely outperforms all other methods for the important subset accuracy metric. This is expected, as recurrent classifier chains are designed to learn the label dependencies required to optimize subset accuracy.

Interestingly, the Robust-RCC also nearly always outperforms the others on *macro* and *micro F1* metrics. Micro F1 is a measure of classification performance for each label *individually*, and not the label set as a whole. However, if the subset accuracy is very high, then it is expected to be high because a high number of exactly-right label sets implies a high number of individually-correct class predictions. The macro F1 is a measure of how well classification performed per-class and having a very high subset accuracy likewise implies a good macro F1 score by a similar argument. Of note is that the performances of the Robust-RCC and the other PU methods do not monotonically increase as the percentage of labeled data increases. This looks surprising at first, but fits the observations previously made about PU methods [31]. Namely, it has been shown they perform better on incompletely-labeled data, as unlabeled points can act to regularize the model [31].

The next-best performing methods are SMiLE [41] and RankPU, depending on the metric. SMiLE is perhaps the most similar method to the Robust-RCC in intent, as it aims to enforce class correlations during training. However, SMiLE enforces the *unconditioned* marginal class correlations rather than the joint probability of the labels conditioned on the input. This difference gives the Robust-RCC an edge in classification performance. RankPU is similar to the Robust-RCC in that it is a Positive-Unlabeled method, although it does not *explicitly* encourage label correlations. RankPU outperforms the Robust-RCC in macro and micro F1 score in a few cases and generally has a high score for these metrics. Despite this, RankPU has a *low* subset accuracy score. This fits with prior work that shows that a high score on multi-label metrics such as the F-1 scores does *not* imply that label correlations are being learned, as learning the label correlations would imply a high subset accuracy [10]. Other than the baseline RCC model, the *CleanLab* classifiers are the worst performing, likely because *CleanLab drops* instances from the training data that it identifies as unlabeled positives. This significantly decreases the amount of training data when a large proportion of class instances are unlabeled, as is the case in this experiment. This indicates that merely *dropping* likely unlabeled positive instances is not a viable solution in this setting due to the fact that nearly every instance will likely have some class unlabeled.

6.5 Ablation Study: Order Free Component

To understand the effect of the order-free classification on the Robust-RCC's performance, we also perform an ablation study comparing the Robust-RCC with two common label prediction orders, frequent-to-rare (FTR) and rare-to-frequent (RTF), on PASCAL VOC 2007. As expected, Figure 3 shows that order-free outperforms these preset orderings. Second-best is frequent-to-rare, indicating that it may be better for the Robust-RCC to predict the "easier" classes before the "harder" classes in most cases. Additionally, the order-free predictions have lower variance than the ordered predictions. This implies that allowing learnable orderings does indeed mitigate the challenge of error propagation during label prediction.

7 CONCLUSION

In this work, we introduce Robust-RCC, the first approach for training RCCs given multi-label data with missing labels. To achieve this, we introduce the multi-incomplete-label risk (MILR), a novel formulation of the multi-label risk that we prove can safely be computed from incompletely labeled data. With MILR, we succeed to train a recurrent classifier chain to match the distribution of the true fully-labeled data, despite access to only incomplete labels. Using three multi-label datasets, we conclusively demonstrate that our approach outperforms all major state-of-the-art alternatives on four common metrics. Our approach takes a large step forward for multi-label classification by RCCs to be applicable even in domains where fully labeled data is not available.

8 ACKNOWLEDGEMENTS

This research was supported in part by grants from DARPA WASH program HR001117S0032 and U.S. Dept. of Education P200A150306. Results in this paper were obtained in part using a computing system acquired through NSF MRI grant DMS-1337943 to WPI.

REFERENCES

[1] A. H. Akbarnejad and M. S. Baghshah. 2019. An Efficient Semi-Supervised Multi-label Classifier Capable of Handling Missing Labels. *IEEE TKDE* (2019).

[2] Jessa Bekker and Jesse Davis. 2018. Estimating the class prior in positive and unlabeled data through decision tree induction. In *AAAI*.

[3] Jessa Bekker and Jesse Davis. 2020. Learning from positive and unlabeled data: A survey. *Machine Learning* (2020).

[4] Jessa Bekker, Pieter Robberechts, and Jesse Davis. 2019. Beyond the selected completely at random assumption for learning from positive and unlabeled data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer.

[5] Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. 2004. Learning multi-label scene classification. *Pattern Recognition* (2004).

[6] Serhat Selcuk Bucak, Rong Jin, and Anil K Jain. 2011. Multi-label learning with incomplete class assignments. In *CVPR*.

[7] Shiyu Chang, Yang Zhang, Jiliang Tang, Dawei Yin, Yi Chang, Mark A Hasegawa-Johnson, and Thomas S Huang. 2016. Positive-unlabeled learning in streaming networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*.

[8] Shang-Fu Chen, Yi-Chen Chen, Chih-Kuan Yeh, and Yu-Chiang Frank Wang. 2018. Order-free rnn with visual attention for multi-label classification. In *AAAI*.

[9] Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. 2019. Multi-label image recognition with graph convolutional networks. In *CVPR*.

[10] Weiwei Cheng, Eyke Hüllermeier, and Krzysztof J Dembczynski. 2010. Bayes optimal multilabel classification via probabilistic classifier chains. In *ICML*.

[11] Hong-Min Chu, Chih-Kuan Yeh, and Yu-Chiang Frank Wang. 2018. Deep generative models for weakly-supervised multi-label classification. In *ECCV*.

[12] Elijah Cole, Oisin Mac Aodha, Titouan Lorieul, Pietro Perona, Dan Morris, and Nebojsa Jojic. 2021. Multi-Label Learning from Single Positive Labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

[14] Marthinus Du Plessis, Gang Niu, and Masashi Sugiyama. 2015. Convex formulation for learning from positive and unlabeled data. In *ICML*.

[15] Pinar Duygulu, Kobus Barnard, Joao FG de Freitas, and David A Forsyth. 2002. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *ECCV*.

[16] Charles Elkan and Keith Noto. 2008. Learning classifiers from only positive and unlabeled data. In *ACM SIGKDD*.

[17] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. [n. d.]. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results.

[18] Walter Gerych, Luke Buquicchio, Kavin Chandrasekaran, Abdulaziz Alajaji, Hamid Mansoor, Aidan Murphy, Elke Rundensteiner, and Emmanuel Agu. 2020. BurstPU: Classification of Weakly Labeled Datasets with Sequential Bias. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE.

[19] Walter Gerych, Tom Hartvigsen, Luke Buquicchio, Emmanuel Agu, and Elke Rundensteiner. 2021. Recurrent Bayesian Classifier Chains for Exact Multi-Label Classification. *Advances in Neural Information Processing Systems* 34 (2021).

[20] Thomas Hartvigsen, Cansu Sen, Xiangnan Kong, and Elke Rundensteiner. 2020. Recurrent halting chain for early multi-label classification. In *KDD*.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

[22] Shantanu Jain, Justin Delano, Himanshu Sharma, and Predrag Radivojac. 2020. Class Prior Estimation with Biased Positives and Unlabeled Examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[23] Atsushi Kanehira and Tatsuya Harada. 2016. Multi-label ranking from positive and unlabeled data. In *CVPR*. 5138–5146.

[24] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv* (2014).

[25] Ryuichi Kiryo, Gang Niu, Marthinus C Du Plessis, and Masashi Sugiyama. 2017. Positive-unlabeled learning with non-negative risk estimator. *Advances in neural information processing systems* (2017).

[26] Xiangnan Kong, Michael K Ng, and Zhi-Hua Zhou. 2011. Transductive multilabel learning via label set propagation. *IEEE Transactions on Knowledge and Data Engineering* 25, 3 (2011), 704–719.

[27] Xiangnan Kong, Zhaoming Wu, Li-Jia Li, Ruofei Zhang, Philip S Yu, Hang Wu, and Wei Fan. 2014. Large-scale multi-label learning with incomplete label assignments. In *SDM*.

[28] Weiwei Liu, Haobo Wang, Xiaobo Shen, and Ivor Tsang. 2021. The emerging trends of multi-label learning. *IEEE transactions on pattern analysis and machine intelligence* (2021).

[29] Yi Liu, Rong Jin, and Liu Yang. 2006. Semi-supervised multi-label learning by constrained non-negative matrix factorization. In *AAAi*, Vol. 6. 421–426.

[30] Jinseok Nam, Eneldo Loza Mencía, Hyunwoo J Kim, and Johannes Fürnkranz. 2017. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *NeurIPS*.

[31] Gang Niu, Marthinus Christoffel du Plessis, Tomoya Sakai, Yao Ma, and Masashi Sugiyama. 2016. Theoretical comparisons of positive-unlabeled learning against positive-negative learning. In *NeurIPS*.

[32] Curtis G Northcutt, Lu Jiang, and Isaac L Chuang. 2019. Confident learning: Estimating uncertainty in dataset labels. *arXiv* (2019).

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*.

[34] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2009. Classifier chains for multi-label classification. In *ECML PKDD*.

[35] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2019. Classifier Chains: A Review and Perspectives. *arXiv* (2019).

[36] Emanuele Sansone, Francesco GB De Natale, and Zhi-Hua Zhou. 2018. Efficient training for positive unlabeled learning. *IEEE transactions on pattern analysis and machine intelligence* (2018).

[37] Pierre Stock and Moustapha Cisse. 2018. Convnets and imagenet beyond accuracy: Understanding mistakes and uncovering biases. In *ECCV*.

[38] Guangxin Su, Weitong Chen, and Miao Xu. 2021. Positive-unlabeled learning from imbalanced data. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence, Virtual Event*.

[39] Lijuan Sun, Songhe Feng, Gengyu Lyu, and Congyan Lang. 2019. Robust semi-supervised multi-label learning by triple low-rank regularization. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 269–280.

[40] Yu-Yin Sun, Yin Zhang, and Zhi-Hua Zhou. 2010. Multi-label learning with weak label. In *Twenty-fourth AAAI conference on artificial intelligence*.

[41] Qiaoyu Tan, Yanming Yu, Guoxian Yu, and Jun Wang. 2017. Semi-supervised multi-label classification using incomplete label information. *Neurocomputing* (2017).

[42] Paweł Teisseyre. 2021. Classifier chains for positive unlabelled multi-label learning. *Knowledge-Based Systems* (2021).

[43] C.-P. Tsai and H.-Y. Lee. 2020. Order-free Learning Alleviating Exposure Bias in Multi-label Classification. In *AAAI*.

[44] Jesper E Van Engelen and Holger H Hoos. 2020. A survey on semi-supervised learning. *Machine Learning* (2020).

[45] Haobo Wang, Zhao Li, Jiaming Huang, Pengrui Hui, Weiwei Liu, Tianlei Hu, and Gang Chen. 2021. Collaboration based multi-label propagation for fraud detection. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*. 2477–2483.

[46] Lichen Wang, Yunyu Liu, Can Qin, Gan Sun, and Yun Fu. 2020. Dual Relation Semi-Supervised Multi-Label Learning.. In *AAAI*.

[47] Ya Wang, Dongliang He, Fu Li, Xiang Long, Zhichao Zhou, Jinwen Ma, and Shilei Wen. 2020. Multi-label classification with label graph superimposing. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[48] Baoyuan Wu, Fan Jia, Wei Liu, Bernard Ghanem, and Siwei Lyu. 2018. Multi-label learning with missing labels using mixed dependency graphs. *International Journal of Computer Vision* (2018).

[49] Baoyuan Wu, Zhilei Liu, Shangfei Wang, Bao-Gang Hu, and Qiang Ji. 2014. Multi-label learning with missing labels. In *2014 22nd International Conference on Pattern Recognition*. IEEE.

[50] Baoyuan Wu, Siwei Lyu, Bao-Gang Hu, and Qiang Ji. 2015. Multi-label learning with missing labels for image annotation and facial action unit recognition. *Pattern Recognition* (2015).

[51] Miao Xu, Rong Jin, and Zhi-Hua Zhou. 2013. Speedup matrix completion with side information: Application to multi-label learning. *Advances in neural information processing systems* (2013).

[52] Miao Xu, Yu-Feng Li, and Zhi-Hua Zhou. 2019. Robust multi-label learning with PRO loss. *IEEE TKDE* (2019).

[53] Hao Yang, Joey Tianyi Zhou, and Jianfei Cai. 2016. Improving multi-label learning with missing labels by structured semantic correlations. In *European conference on computer vision*. Springer.

[54] Vacit Oguz Yazici, Abel Gonzalez-Garcia, Arnau Ramisa, Bartlomiej Twardowski, and Joost van de Weijer. 2020. Orderless Recurrent Models for Multi-label Classification. In *CVPR*.

[55] Chih-Kuan Yeh, Wei-Chieh Wu, Wei-Jen Ko, and Yu-Chiang Frank Wang. 2017. Learning deep latent space for multi-label classification. In *Thirty-first AAAI conference on artificial intelligence*.

[56] Hsiang-Fu Yu, Hsin-Yuan Huang, Inderjit Dhillon, and Chih-Jen Lin. 2017. A unified algorithm for one-cass structured matrix factorization with side information. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[57] Yao Zhou, Jianpeng Xu, Jun Wu, Zeinab Taghavi, Evren Korpeoglu, Kannan Achan, and Jingrui He. 2021. PURE: Positive-Unlabeled Recommendation with Generative Adversarial Network. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*.

[58] Yue Zhu, James T Kwok, and Zhi-Hua Zhou. 2017. Multi-label learning with global and local label correlation. *IEEE TKDE* (2017).