IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 22, 2021, accepted February 4, 2022, date of publication February 11, 2022, date of current version February 22, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3151048

MLCM: Multi-Label Confusion Matrix

MOHAMMADREZA HEYDARIAN[1], THOMAS E. DOYLE[1,2,3], (Senior Member, IEEE), AND REZA SAMAVI[3,4], (Senior Member, IEEE)

[1]Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON L8S 4L8, Canada
[2]School of Biomedical Engineering, McMaster University, Hamilton, ON L8S 4L8, Canada
[3]Vector Institute for Artificial Intelligence, Toronto, ON M5G 1M1, Canada
[4]Department of Electrical and Computer Engineering, Ryerson University, Toronto, ON M5B 2K3, Canada

Corresponding author: Thomas E. Doyle (doylet@mcmaster.ca)

This work was supported by the Canadian Department of National Defence Innovation for Defence Excellence and Security (IDEaS) under Award CFPMN2-17.

ABSTRACT Concise and unambiguous assessment of a machine learning algorithm is key to classifier design and performance improvement. In the *multi-class* classification task, where each instance can only be labeled as one class, the confusion matrix is a powerful tool for performance assessment by quantifying the classification overlap. However, in the *multi-label* classification task, where each instance can be labeled with more than one class, the confusion matrix is undefined. Performance assessment of the multi-label classifier is currently based on calculating performance averages, such as hamming loss, precision, recall, and F-score. While the current assessment techniques present a reasonable representation of each class and overall performance, their aggregate nature results in ambiguity when identifying false negative (*FN*) and false positive (*FP*) results. To address this gap, we define a method of creating the *multi-label confusion matrix* (MLCM) based on three proposed categories of multi-label problems. After establishing the shortcomings of current methods for identifying *FN* and *FP*, we demonstrate the usage of the MLCM with the classification of two publicly available multi-label data sets: i) a 12-lead ECG data set with nine classes, and ii) a movie poster data set with eighteen classes. A comparison of the MLCM results against statistics from the current techniques is presented to show the effectiveness in providing a concise and unambiguous understanding of a multi-label classifier behavior.

INDEX TERMS Classification performance, confusion matrix, machine learning, multi-class, multi-label.

I. INTRODUCTION

In the process of developing a machine learning algorithm for classification, researchers often develop several algorithms to assess and compare for the best performance. Each classification method is dependent on many factors such as the data set properties and hyper-parameters. To perform the classification task, based on the total number of classes in the data set and the number of assigned labels to each data instance, we use a binary, multi-class, or multi-label classifier. For evaluation of the classifiers, machine learning developers require an analytical tool that calculates sufficient statistical information on the results. Except for the binary case, the overall accuracy or performance metrics of a classifier offer incomplete insight for refining a model. Such insight is critical with multi-class and multi-label data set modelling.

Many different multi-label classification algorithms have been developed for problems such as text categorization [1], multimedia content annotation [2], disease recognition [3], and web mining [4]. A powerful method for analyzing a multi-class classifier is the 2-dimensional *confusion matrix* to show the distribution of false predictions in one view. The extracted metrics from the confusion matrix, such as *precision*, *recall*, and F^{β} *score* for each class and *micro*, *macro*, and *weighted* average of all classes, are used for measuring the overall performance of a classifier. However, in multi-label classification problems, the confusion matrix is an undefined method. In such problems the performance measurement is limited to calculating aggregate metrics [5] such as *hamming loss*, *accuracy*, *subset accuracy*, *precision*, *recall*, and F^{β} *score* (F-score). This limitation introduces ambiguity on number of false negative (*FN*) or false positive (*FP*) associated with each class.

Sokolova and Lapalme [6] presented a systematic analysis of twenty-four performance metrics for evaluation of binary, multi-class, multi-label, and hierarchical learning

The associate editor coordinating the review of this manuscript and approving it for publication was Chao-Yang Chen.

The MLCM code can be found at https://github.com/mrh110/mlcm

approaches. Tsoumakas *et al.* [7] presented the various evaluation methods for multi-label approaches by categorizing them to bipartitions and rankings methods. In other work, Zhang and Zhou [5] have categorized and described existing evaluation metrics followed by categorizing and describing classification approaches for specifically multi-label problems.

Pereira *et al.* [8] analyzed the correlation between performance measurement methods to identify the most frequent and popular evaluation methods used in literature based on research from Spolaor *et al.* [9]. Based on their study, hamming loss, accuracy, and F-score are the most frequent measurement tools that were used to report the performance of a multi-label classifier.

The aggregate metrics for multi-label classification provide the overall performance of a classifier, which is adequate to evaluate and compare the classifiers, but misses any details of the distribution of incorrectly classified instances. These methods are able to report the statistics on each class independently, but miss the specifics of incorrect prediction distribution over other classes. Creating a multi-label confusion matrix would help the developers of these classifiers, in the design phase, to improve the overall metrics in their reports. The confusion matrix could also be used as a weighting matrix for the cost sensitive loss metrics [10], especially for imbalanced data sets.

The commonly used method in the Python programming language library[1] for calculating *FN* only determines if a true label is not predicted correctly and, similarly, for calculating *FP* only determines if a non-assigned label is predicted incorrectly, without reporting how many incorrect predictions occurred and how many true labels were not predicted for the associated instance. For example, if we consider a multi-label data set with three classes, as shown in Table 1, for all instances (except for instances 1 and 8), the calculated *FN* and *FP* by the current method [5] are ambiguous. We report on two problematic cases: i) instance 2 was labeled with all three classes, and the classifier was unable to predict label C_1 while there is no incorrect prediction to be *FN* of class C_1, ii) instance 5 was labeled with classes C_0 and C_1 which were predicted and are counted as true positive (*TP*), but label C_2 was also predicted where none of the true labels is missing to be assigned for this incorrect prediction.

The current method reports the results based on a *one-vs-rest* that is calculated *FN*, *FP*, *TP*, and true negative (*TN*) of one class versus all other classes. For instances in Table 1, three one-vs-rest confusion matrices are created. Tables 2a, 2b, and 2c show the resulting one-vs-rest confusion matrices for classes C_0, C_1, and C_2, respectively (using the `scikit-learn` Python library (`sklearn`)). Table 2d shows the element wise summation of these three matrices which is used for calculating micro average of precision, recall, and F-score. In Table 2d, we observe that the summation of *FN* (e.g., 5) is not equal to the summation of *FP* (e.g., 9) for all classes, where the assertion is that each *FN* to a class is a *FP* to another class. For example, if the true label is C_1 and the predicted label is C_2, there is one *FN* for class C_1 which is one *FP* to class C_2. This simple example shows that the current method for identifying *FP* and *FN* is incomplete by only focusing on one class at a time and ignoring the combination of the true and predicted labels for other classes.

[1]Reported by Zhang and Zhou [5], implemented as part of the `scikit-learn` Python library by Pedregosa *et al.* [11], as part of the `Mulan` Java API by Tsoumakas *et al.* [12], and as part of `MEKA` Java API by Read *et al.* [13].

TABLE 1. Example of true and predicted labels.

instance #	category	True Labels			Predicted Labels		
		C_0	C_1	C_2	C_0	C_1	C_2
1	1	1	1	0	1	1	0
2	1	1	1	1	1	0	1
3	1	0	0	0	0	0	0
4	2	1	0	0	1	1	1
5	2	1	1	0	1	1	1
6	2	0	0	0	0	1	1
7	3	1	0	0	0	1	1
8	3	1	1	0	1	0	1
9	3	1	1	0	0	0	1

TABLE 2. One-vs-rest confusion matrix for sample instances in Table 1 using sklearn library. First and second rows show [TN, FP] and [FN, TP], respectively.

(a) C_0

2	0
2	5

(b) C_1

1	3
3	2

(c) C_2

2	6
0	1

(d) Sum

5	9
5	8

In this paper, we propose a novel method that defines the confusion matrix for the multi-label classifiers (MLCM) that retains the familiar structure of the multi-class confusion matrix and satisfies the requirements of a 2-dimensional confusion matrix. Our method is concise and unambiguous for all combinations of true and predicted labels while properly accounting for all cases. The proposed MLCM considers all possible combinations of true and predicted labels, which results in accurately extracting *FN*, *FP*, *TP*, and *TN* for further statistical calculation such as precision, recall, and F-score.

This paper is organized as follows. Section 2 provides an overview of related common evaluation methods for multi-label classification algorithms. Section 3 gives a detailed description of our proposed method for creating the confusion matrix for a multi-label classifier. Section 4 presents the results of applying MLCM on two data sets. Section 5 summarizes the advantages of MLCM for analyzing the performance of a multi-label classifier.

II. BACKGROUND

In the last two decades, there have been many works dealing with multi-label data sets. These data sets are from different domains, such as text, graphics, waveforms, and medical images. For clarity and consistency, we define the

TABLE 3. Classification categories.

Classifier Category	# of classes	# of labels per instance	Example
Binary	2	1	Black&White or Colour Movie
Multi-Class	$q > 2$	1	Movie Type (Film, TV Movie, TV Miniseries, TV Series, Video)
Multi-Label	$q \geq 2$	$0, 1, \ldots, q$	Movie Genre ('Comedy', 'Drama', Romance', 'Comedy&Drama', Comedy&Drama&Romace', etc.)

categories Binary, Multi-Class, and Multi-Label classification in Table 3. There is no similar work to our proposed MLCM, thus we review some of the related works. First, we highlight review papers that have covered performance metrics for multi-label classifiers.

Sokolova and Lapalme [6] presented a systematic analysis of twenty-four performance metrics used for evaluation of the binary, multi-class, multi-label, and hierarchical learning approaches. They assessed measure invariance by analyzing the type of changes that do not change a measure's value (e.g., changes to *TN* does not change the precision or recall). They reviewed four metrics for multi-label classification (exact match ratio, labeling F-score, retrieval F-score, and hamming loss).

Dembczyński *et al.* [14] investigated conditional and marginal (unconditional) dependencies between the labels for multi-label classification. They presented three scenarios where exploiting one of those dependencies may boost the predictive performance of a classifier, which is also dependent on the type of loss function (e.g., hamming loss and subset 0/1 loss) to be minimized. The scenarios that they investigated were individual label view, joint label view, and joint distribution view.

Zhang and Zhou [5] categorized the classification methods for multi-label data sets into two groups. The first group of methods named *Problem Transformation Methods* which includes *binary relevance*, *classifier chains*, *calibrated label ranking*, and *random K-label sets*. Generally, these methods focus on transferring the data set, including its labeling, to a shape that could be manipulated as a binary-class or multi-class problem. The second group of methods named *Algorithm Adaptation Methods* which includes *multi-label K-NN*, *multi-label decision tree*, *ranking SVM*, and *collective multi-label classifier*. Generally, these methods focus on adapting multi-class classifier algorithms to apply on multi-label data sets.

Zhang and Zhou [5] focused on performance metrics used for evaluation of multi-label classification approaches. They categorized the evaluation tools to *example-based* and *label-based* metrics, where for each of these two categories, if the intermediate real-valued probability of labels is available, a new set of ranking metrics was defined. Example-based metrics include *subset-ccuracy*, *hamming loss*, and *example-based- accuracy*, *precision*, *recall*, and F^{β} *score*. When intermediate probability value is available, four example-based ranking metrics can be defined as *one-error*, *coverage*, *ranking-loss*, and *average-precision*. Label-based metrics include *micro* and *macro averaging* of *accuracy*, *precision*, *recall*, and F^{β} *score*. When intermediate probability value is available, *AUC* (Area Under Curve) can be defined as a label-based ranking metric.

Waegeman *et al.* [15] provided an analysis of different approaches such as hamming loss and subset 0/1 loss for maximizing the F-measure. Koyejo *et al.* [16] proposed a framework for the analysis of multi-label classification metrics based on average binary performance metrics. These two works used the same method reported in [5] for calculating *TP*, *TN*, *FP*, and *FN* as the fundamental metrics for their research on characterizing the Bayes optimal classifier for multi-label metrics.

Several researchers, noted below, have reported on the improvement of the multi-label classification algorithms while using existing overall performance metrics for comparing the methods.

Madjarov *et al.* [17] compared 12 multi-label learning methods using 16 evaluation metrics over 11 benchmark data sets. They used six example-based, six label-based, and four ranking-based evaluation metrics, all reporting the overall performance measure of classifiers. Their goal was to show the best-performing methods for multi-label classification.

Lee *et al.* [18] proposed a multi-label learning method inspired by the classifier chain approach. They used four overall evaluation metrics (exact match loss, hamming loss, Jaccard index, and macro F1-score) to compare their proposed method with the state-of-the-art approaches.

Wu and Zhou [19], based on their proposed concepts of label-wise/instance-wise margins, showed that some performance metrics are to be optimized by label-wise effective classifiers, whereas some by instance-wise effective classifiers. They also concluded that there are some shared properties among different subsets of multi-label performance metrics which explains why some metrics seem to be redundant in experiments.

Eftimov and Kocev [20] proposed a performance metrics fusion approach based on multi-criteria decision analysis, which all are overall performance metrics of a classifier. They used their previous experimental study [17] to investigate the performance of their proposed methodology.

Bogaert *et al.* [21] evaluated multi-label classification techniques and recommender systems for cross-sell purposes in the financial services sector. They used five overall evaluation metrics with more focus on the harmonic and geometric mean of precision and recall in order to evaluate the performance of the analytical techniques.

Menon *et al.* [22] employed reduction of Multi-label problem to a suitable series of binary or multi-class classification problems. They used precision and recall to compare the reduction approaches. They showed that each reduction is provably consistent with respect to either precision or recall, but not both, so in general no reduction can be optimal for both metrics.

There are several different metrics for evaluation of a multi-label classifier, however, some of these metrics are highly correlated (e.g., F1-score is correlated to precision and recall) as reported by Pereira *et al.* [8] and reporting them together will yield little, if any, additional information regarding the performance of a classifier. Spolaor *et al.* [9] reported that the hamming loss, accuracy, and F-score are the most frequent metrics that were used to report the performance of a multi-label classifier in 55, 26, and 18 papers, respectively, out of 64 papers published on multi-label classification.

A. PERFORMANCE METRICS

The current popular methods for reporting the performance of a multi-label classifier include the hamming loss, precision, recall, and F-score for each class with the micro, macro, and weighted average of these scores from all classes as demonstrated in Table 11. The caveat of these measurement methods is that these statistics only relate to the *overall* aggregated performance of the classifier, unable to provide the distribution of *FN* or *FP* prediction on other classes.

Hamming loss is the symmetric difference between the set of true labels and the set of predicted labels which may be calculated by applying the exclusive disjunction function for each instance and then adding them together to find the overall measurement as illustrated in (1),

$$hammingloss = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{q}\Delta(T_i, P_i) \tag{1}$$

where m is the number of instances, T_i and P_i are the list of true and predicted labels for instance i, respectively, q is the number of predefined classes, and Δ is the exclusive disjunction function.

Equations (2), (3), and (4) show the precision, recall, and F-score for class C, respectively,

$$precision_c = \frac{TP_c}{TP_c + FP_c} \tag{2}$$

$$recall_c = \frac{TP_c}{TP_c + FN_c} \tag{3}$$

$$F^{\beta}score_c = \frac{(\beta^2+1)TP_c}{(\beta^2+1)TP_c + \beta^2 FN_c + FP_c} \tag{4}$$

where TP_c, FP_c, and FN_c are the number of *TP*, *FP*, and *FN* predictions of the classifier for class C, respectively. In measuring $F^{\beta}score$, β is the balancing factor with positive amount which commonly is set to 1 (e.g., F1-score), results in the harmonic mean of precision and recall.

For an overall analysis of a classifier, we can use *micro*, *macro*, and *weighted* average of precision, recall, and F-score. Equations (5) to (7) show the formula for calculating these statistics for $F^{\beta}score$,

$$F^{\beta}_{micro}score = \frac{(\beta^2+1)\sum_{c=0}^{q-1} TP_c}{(\beta^2+1)\sum_{c=0}^{q-1} TP_c + \beta^2\sum_{c=0}^{q-1} FN_c + \sum_{c=0}^{q-1} FP_c} \tag{5}$$

$$F^{\beta}_{macro}score = \frac{1}{q}\sum_{c=0}^{q-1} F^{\beta}_c \tag{6}$$

$$F^{\beta}_{weighted}score = \frac{\sum_{c=0}^{q-1} F^{\beta}_c S_c}{\sum_{c=0}^{q-1} S_c} \tag{7}$$

where q is the number of classes, F^{β}_c is the $F^{\beta}score$ of class C calculated from (4), and S_c (e.g., Support column in Table 11) is the number of the instances that have been labeled as class C. Equations for calculating micro, macro, and weighted average of precision and recall are similar to (5) to (7), respectively.

B. MULTI-CLASS CONFUSION MATRIX

A confusion matrix is a two-dimensional matrix where its rows show the true labels and its columns show the predicted labels by a classifier. Creating a confusion matrix for a multi-class classifier is simple and straightforward as shown in (8),

$$\mathbf{M}(r, c) = \sum_{i=1}^{m}(I(y_i = r)\, I(h(x_i) = c)), \quad \forall r, c \in \{0, \ldots, q-1\} \tag{8}$$

where $\mathbf{M}$ is the confusion matrix, r and c are the row and column of the confusion matrix, m is the number of instances in the test data set, $I(.)$ is the indicator function, x_i is the i^{th} input to classifier $h(.)$, y_i is the true label assigned to input x_i, and q is the number of classes. Algorithm 1 shows the method for creating the confusion matrix for a multi-class classifier.

Algorithm 1 Multi-Class Confusion Matrix

```
for each input instance do
    r = assigned label
    c = predicted label
    M(r, c) += 1
end for
```

A confusion matrix shows the distribution of predictions over all classes in one concise view. By row-wise normalizing of the confusion matrix, the percentage of *FN* for each class corresponding to each row is obtained. The cells on the main diagonal of the normalized confusion matrix show the recall for the corresponding class. While the normalized matrix is useful for finding the percentage of the true and the false prediction, the confusion matrix with the actual number of counting the true and the false prediction will give the information on the size of each class in the case of imbalanced data sets.

C. MULTI-LABEL PERFORMANCE MEASUREMENTS

In the multi-label data set, for the instances that were assigned with only one true label and the classifier predicted only one label, we could use (8) for creating a confusion matrix. But (8) is unable to handle other combinations of true and predicted

labels for multi-label data sets. For example, this equation is not appropriate for any of the instances shown in Table 1.

To the best of our knowledge, there exists no complete method in practice or literature to create a multi-label confusion matrix.

While the main benefit of having a confusion matrix is to visualize and analyze the distribution and overlap of correctly and incorrectly predicted labels over other labels in one view, it also is used to calculate *TP*, *TN*, *FP*, and *FN* to be used in the calculation of precision, recall, and F-score as the most informative and frequently used metrics in the assessment of classifiers for each class.

The `sklearn` Python library [11] has a function called `multilabel_confusion_matrix` which, despite the name of the function, is a one-vs-rest confusion matrix (similar to the binary confusion matrix). This means that `sklearn`'s `multilabel_confusion_matrix` function calculates *TP*, *TN*, *FP*, and *FN* for each class (using the method reported by [5]) rather than creating a 2-dimensional q-class confusion matrix. This approach only checks one label at a time and ignores the situation of other true and predicted labels in the data instance resulting in an incomplete and potentially skewed view of the classifier performance. For example, for two distinctly different cases, the calculated precision and recall for the instance 1 of Table 1 are exactly equal to the calculated precision and recall for the instance 5 of Table 1 (e.g., 1.00, 1.00, and 0.00 for classes C_0, C_1 and C_2, respectively). Instances 1 and 5 were assigned with the same true labels. There is no incorrect prediction for the instance 1 and the calculated precision and recall are correct. On the other hand, there is one incorrect prediction for the instance 5 where the `sklearn` is unable to consider it and returns the same precision and recall as for the instance 1. A precise method should consider the incorrect prediction for instance 5 (e.g., *FN* to classes C_0 and C_1) and therefore the recall should be smaller than 1.00 for classes C_0 and C_1 (e.g., 0.50, 0.50, and 0.00 for classes C_0, C_1 and C_2, respectively).

The implementation of the `sklearn` library for calculating *FN* is limited because it only checks if a true label was not predicted correctly without considering how many incorrect predictions occurred. Similarly, the method for calculating *FP* only checks if a non-assigned label was predicted incorrectly without considering how many true labels were not predicted. The independent counting of *FN* and *FP* by this method results that the summation of *FN* for all classes is unequal to the summation of *FP* for all classes. By the definition of the confusion matrix, each *FN* from a class is a *FP* to another class and consequently their summations must be equal. The `sklearn` documentation acknowledges the limitation described here and states that in extending a binary metric to multi-class or multi-label problems, the `sklearn` treats the data as a collection of binary problems, one for each class. This is the main limiting factor in computing the multi-label confusion matrix that this proposal intends to address.

III. MULTI-LABEL CONFUSION MATRIX (MLCM)

In multi-label classification problems, where each instance can be labeled with more than one class, the confusion matrix is an undefined assessment tool. To avoid the ambiguity of true and predicted labels combinations for each instance, we define categories for the handling of each combination of true and predicted labels and based on this categorization we introduce our proposed algorithm for the creation of the MLCM.

Let T_i be the set of true labels assigned to instance i and P_i be the set of predicted labels for that instance. Let divide the set T_i to two subsets T_{i1} and T_{i2} for the set of predicted and not-predicted true labels, respectively. Similarly, let divide the set P_i to two subsets P_{i1} and P_{i2} for the set of correctly predicted and incorrectly predicted labels, respectively. From this definition, we see that T_{i1} is equal to P_{i1}.

A. NO-LABEL ASSIGNED OR PREDICTED

For labeling and classification, the occurrence of no-label-assigned or no-label-predicted must be acknowledged and recorded. Consider for multi-label classifiers, the output nodes (e.g., from the last layer of a Neural Network) with the probabilities higher than a predefined threshold (e.g., 0.50) would be reported as the predicted labels. It is possible that some of the output nodes corresponding to true labels are smaller than the predefined probability threshold and therefore would not be predicted (e.g., class C_1 of instance 2 in Table 1). For this situation, while there is no incorrect prediction (e.g., only two genres from four true genres of a movie were predicted), the number of predicted labels is less than the number of true labels (e.g., could be no label predicted at all). Definition 1 summarizes this situation. There is also a possibility that none of the predefined classes were assigned to an instance of a data set (e.g., none of the predefined genres assigned to a movie). Definition 2 summarizes this situation.

Definition 1: For instance i of a data set, *NPL* (No Predicted Label) is a situation in the combination of true and predicted labels where one or more true labels are not predicted while there is no incorrect prediction. Based on the definition of T_i and P_i, this is the case that $T_{i2} \neq \varnothing$ and $P_{i2} = \varnothing$.

Definition 2: For instance i of a data set, *NTL* (No True Label) is a situation in the combination of true and predicted labels where there is no true label assigned to instance i of a data set. Based on the definition of T_i and P_i, this is the case that $T_i = \varnothing$.

For the multi-class confusion matrix, there exists one row and one column for each predefined class. For MLCM, in addition to one row and one column for each predefined class, we add one row (i.e., last row *NTL*) to the confusion matrix to capture the possibility that none of the true labels were assigned. We also add one column (i.e., last column *NPL*) to the confusion matrix to show the situation of no prediction for some (or all) of the true labels. Therefore, for a

classification task with q classes, MLCM has $q+1$ rows and $q+1$ columns in total. Rows and columns from 0 to $q-1$ are used for classes C_0 to C_{q-1}, where the row q and the column q are used for the no label assigned and no label predicted situations, respectively.

B. COMBINATIONS OF TRUE AND PREDICTED LABELS

We formally define the categories of different combinations of true and predicted labels as follows:

Category One: $P_i \subseteq T_i$
Category Two: $T_i \subset P_i$
Category Three: $T_{i2} \neq \varnothing$ and $P_{i2} \neq \varnothing$

where $\varnothing$ means empty set. Condition $P_i \subseteq T_i$ for category one covers the situations that all or a subset of true labels were predicted correctly and there is not any incorrect prediction (i.e., $P_{i2} = \varnothing$). Condition $T_i \subset P_i$ for category two covers the situations that all true labels were predicted correctly, but there are also some incorrect predictions (i.e., $T_{i2} = \varnothing$ and $P_{i2} \neq \varnothing$). Conditions $T_{i2} \neq \varnothing$ and $P_{i2} \neq \varnothing$ for category three means that there are some true labels that were not predicted and there are some incorrect predictions. Set T_{i1} (and equally P_{i1}) could be empty for the situation that none of the true labels were predicted.

For all three categories, our MLCM algorithm updates the confusion matrix in two steps. At the first step, for all three categories, it increases the main diagonal of the confusion matrix for all labels in T_{i1} to count *TP* as shown in Algorithm 2.

Algorithm 2 First Step of MLCM

```
for r in T_i1 do
    M(r, r) += 1
end for
if T_i = ∅ and P_i = ∅ then
    M(NTL, NPL) += 1
end if
```

Equation (9) shows the first step of updates to MLCM,

$$\mathbf{M}(r,r) = \sum_{i=1}^{m} (I(y_{i,r} = 1)\, I(h_r(x_i) = 1)), \quad \forall r \in \{0, \ldots, q-1\} \tag{9}$$

where $y_{i,r}$ shows the occurrence of the true label r (i.e., class C_r) for input x_i (i.e., 1 for assigning label r and 0 for not assigning label r), $h_r(x_i)$ is the prediction for label r of input x_i, and the rest of the parameters are the same as parameters defined in (8). The situation of $P_i = \varnothing$ and $T_i = \varnothing$ cannot be handled using (9) as counting *TP* is based on value 1 for each label of instances, but in this situation the algorithm need to deal with value 0 for *TP*. Thus the algorithm needs to check the situation separately using (10):

$$\mathbf{M}(NTL, NPL) = \sum_{i=1}^{m} 1, \quad T_i = \varnothing;\ P_i = \varnothing. \tag{10}$$

Then, at the second step, it updates the MLCM for counting the *FN* and *FP* which the methods are different based on the three categories. Checking the conditions for belonging to these categories are excluded in the equations. Parameters used in the following equations were defined in the previous equations.

To illustrate our proposed algorithm, we use an example set of nine instances with true and predicted labels. This example will demonstrate our proposed algorithm and how our approach handles ambiguities. Table 1 shows the true labels assigned to the instances of a data set and the predicted labels by the classifier corresponding to each of the instances. For simplicity of illustration, a maximum of three classes was applied. Table 4 shows the results of applying MLCM algorithm to each of the instances (i.e., Tables 4a to 4i for instances 1 to 9, respectively) and shows the final MLCM in Table 4j by combining the MLCM from each of the instances.

1) CATEGORY ONE

Equation (11) and Algorithm 3 show the second step of updates to MLCM for category one.

$$\mathbf{M}(r, NPL) = \sum_{i=1}^{m} (I(y_{i,r} = 1)\, I(h_r(x_i) = 0)), \quad \forall r \in \{0, \ldots, q-1\}. \tag{11}$$

Algorithm 3 Category 1 of MLCM

```
for r in T_i2 do
    M(r,NPL) += 1
end for
```

For each label in T_{i2}, the algorithm increases the corresponding element on the *NPL* column of the confusion matrix, meaning that there is a *FN* prediction for each label in T_{i2} to the *NPL*.

For the example shown in Table 1, instances 1 to 3 are in category one. For the instance 1, both true labels (C_0 and C_1) were predicted correctly (i.e., $P_i = T_i$), so the value of element at column C_0 of row C_0 and column C_1 of row C_1 need to be incremented by one to count the *TP*.

For the instance 2, labels C_0 and C_2 were predicted correctly, so the value of elements at column C_0 of row C_0 and column C_2 of row C_2 need to be incremented by one to count the *TP*. There is another true label (i.e., C_1) where there is no more predicted label to assign to it (i.e., $T_{i2} = \{C_1\}$ and $P_{i2} = \varnothing$). This situation is considered as *FN* for class C_1 to the no-label prediction, therefore the value of element at column *NPL* of row C_1 is incremented by one.

For the instance 3, none of the labels were assigned to this instance and none of the labels were predicted for it (i.e., *TP*), thus the value of the element at column *NPL* of row *NTL* is incremented by one. For instances 1 and 3 the set P_i is equal to set T_i, thus $T_{i2} = \varnothing$ and the algorithm terminates before its second step.

TABLE 4. MLCM for instances in Table 1. Rows are true labels for C_0, C_1, C_2, and *NTL*. Columns are predicted labels for C_0, C_1, C_2, and *NPL*.

(a) instance 1

1	0	0	0
0	**1**	0	0
0	0	0	0
0	0	0	0

(b) instance 2

1	0	0	0
0	0	0	1
0	0	**1**	0
0	0	0	0

(c) instance 3

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	**1**

(d) instance 4

1	1	1	0
0	0	0	0
0	0	0	0
0	0	0	0

(e) instance 5

1	0	1	0
0	**1**	1	0
0	0	0	0
0	0	0	0

(f) instance 6

0	0	0	0
0	0	0	0
0	0	0	0
0	1	1	0

(g) instance 7

0	1	1	0
0	0	0	0
0	0	0	0
0	0	0	0

(h) instance 8

1	0	0	0
0	0	1	0
0	0	0	0
0	0	0	0

(i) instance 9

0	0	1	0
0	0	1	0
0	0	0	0
0	0	0	0

(j) total MLCM

5	2	4	0
0	**2**	3	1
0	0	**1**	0
0	1	1	**1**

2) CATEGORY TWO

Equations (12) and (13) and Algorithm 4 show the second step of updates to MLCM for category two.

$$\mathbf{M}(r,c) = \sum_{i=1}^{m} (I(y_{i,r}=1)\, I(h_r(x_i)=1) \times I(y_{i,c}=0)\, I(h_c(x_i)=1)), \quad \forall r,c \in \{0,\ldots,q-1\}. \tag{12}$$

Algorithm 4 Category 2 of MLCM

```
for r in T_i do
  for c in P_i2 do
    M(r, c) + = 1
  end for
end for
if T_i = ∅ then
  for c in P_i2 do
    M(NTL,c) + = 1
  end for
end if
```

For each label in P_{i2}, the algorithm increases the value of elements in corresponding columns related to labels in P_{i2} for all rows corresponding to labels in T_i, meaning that although all true labels were predicted correctly, the incorrect predictions will be counted as *FN* to all of correctly predicted labels since no unpredicted true label exists to be considered for incorrect prediction(s) (i.e., $T_{i2} = \varnothing$). The incremented value of *FN* in the confusion matrix is considered as *FP* for all classes in P_{i2}. The special case of $T_i = \varnothing$ while $P_i \neq \varnothing$ in category two is addressed using (13):

$$\mathbf{M}(NTL,c) = \sum_{i=1}^{m} I(h_c(x_i)=1), \quad \forall c \in \{0,\ldots,q-1\}; \quad T_i = \varnothing;\ P_i \neq \varnothing. \tag{13}$$

For the example shown in Table 1, instances 4, 5, and 6 are in category two. For the instance 4, two labels (i.e. C_1 and C_2) were predicted incorrectly in addition to the correctly predicted true label C_0. In this case, there is one *TP* to class C_0 which will increase the value of the element at column C_0 of row C_0 and then row C_0 needed to be updated for all additional incorrect predicted labels (i.e., increase the value of elements at columns C_1 and C_2 of row C_0). This update to the confusion matrix demonstrates that although the classifier predicted the true label for this instance (one *TP*), it also predicted two additional labels for this instance that should be considered as *FN* for each false prediction.

For the instance 5, two labels were assigned to the instance as the true labels (i.e., C_0 and C_1) and both were predicted correctly, but there is one extra predicted label (i.e., C_2) that was predicted incorrectly (*FN*). In this case, the update to the MLCM will be one *TP* to each of classes C_0 and C_1, and then the value of the element at column C_2 needs to be incremented for both C_0 and C_1 rows to show the possibility of *FN* to both true labels. This will count the *FN* for each true label which is helpful while analyzing each row of the confusion matrix.

For the instance 6, there is no-label assigned but labels C_1 and C_2 were incorrectly predicted. For this situation, the value of element at columns C_1 and C_2 of the last row (i.e., *NTL*) are incremented to show the *FN* of no-class to classes C_1 and C_2. This incremented value of *FN* in the confusion matrix is considered as *FP* to classes C_1 and C_2.

3) CATEGORY THREE

Equation (14) and Algorithm 5 show the second step of updates to MLCM for category three.

$$\mathbf{M}(r,c) = \sum_{i=1}^{m} (I(y_{i,r}=1)\, I(h_r(x_i)=0) \times I(y_{i,c}=0)\, I(h_c(x_i)=1)), \quad \forall r,c \in \{0,\ldots,q-1\}. \tag{14}$$

For each label in P_{i2}, the algorithm increases the corresponding value of elements at columns related to labels in P_{i2} for all rows corresponding to labels in T_{i2}, meaning that the incorrect predictions will be counted as *FN* to all of not-predicted labels (i.e., classes in T_{i2}). This update also counted as *FP* to classes in P_{i2}.

For the example shown in Table 1, instances 7, 8 and 9 are in category three. For the instance 7, the true label is C_0 where labels C_1 and C_2 were predicted incorrectly (i.e., $T_{i1} = \varnothing$ and $T_{i2} = T_i$). The update to the confusion matrix would be two *FN* for C_0 which was scattered over classes C_1 and C_2, also considered as the *FP* for C_1 and C_2 (i.e., increment of the value of elements at columns C_1 and C_2 of row C_0).

For the instances 8 and 9 two labels assigned to each of the instances as the true labels (i.e., C_0 and C_1). For instances 8, one of the labels (C_0) were predicted correctly (i.e., set T_{i1}), so the value of element at column C_0 of row C_0 is incremented as the *TP*; then at the second step of the algorithm, the value of element at column C_2 of row C_1 incremented (i.e., *FN* of class C_1, also *FP* to class C_2). For the instance 9 none of the true labels were predicted correctly (i.e., $T_{i1} = \varnothing$ and $T_{i2} = T_i$) and label C_2 was predicted incorrectly. Thus, the value of elements at column C_2 of rows C_0 and C_1 are incremented.

Algorithm 5 Category 3 of MLCM

```
for r in T_i2 do
  for c in P_i2 do
    M(r, c) += 1
  end for
end for
```

To show an example that the size of sets T_{i2} and P_{i2} are both greater than one, we illustrate an example of an instance with five classes in Table 5. As we see, $T_i = \{C_0, C_1, C_2\}$, $P_i = \{C_0, C_3, C_4\}$, $T_{i1} = \{C_0\}$, $T_{i2} = \{C_1, C_2\}$, $P_{i1} = \{C_0\}$, and $P_{i2} = \{C_3, C_4\}$. Thus the update to MLCM will be one *TP* to class C_0 (i.e., increase the value of element at column C_0 of row C_0) and *FN* for each of classes C_1 and C_2 to classes C_3 and C_4 (i.e., increase the value of elements at columns C_3 and C_4 of rows C_1 and C_2).

TABLE 5. True vs predicted labels for a sample instance with five classes.

	Classes				
sample instance	C_0	C_1	C_2	C_3	C_4
True Label	1	1	1	0	0
Predicted Label	1	**0**	**0**	**1**	**1**

C. INTERPRETING MLCM

We can extract *TP*, *TN*, *FP*, and *FN* of each class from MLCM and consequently compute other statistics such as *precision*, *recall*, and F^β *score* for each of the classes using (2) to (4), respectively. We could compute *micro*, *macro*, and *weighted* average of *precision*, *recall*, and F^β *score* using extracted *TP*, *TN*, *FP*, and *FN* which was explained in section II-A. Table 6 shows how we can extract *TP*, *TN*, *FP*, and *FN* for class C_2 from MLCM.

Generally speaking, for row k ($k \in \{0, \ldots, q\}$) which corresponds to class C_k (or *NTL* for $k = q$), the cell on the main diagonal shows the *TP* to class C_k, where other cells on the row show the *FN* to class C_k. Except for the cell on the main diagonal, the cells on column k show the *FP* to class C_k. The summation of cells on row k, excluding the cell on the main diagonal, and the summation of cells on column k, excluding the cell on the main diagonal, represent the overall *FN* and *FP* to class C_k, respectively.

TABLE 6. Extracting information from MLCM for class C_2.

		Predicted Labels					
	Classes	C_0	C_1	C_2	C_3	C_4	*NPL*
True Labels	C_0	$TN_{2,0}$		$FP_{0,2}$			
	C_1		$TN_{2,1}$	$FP_{1,2}$			
	C_2	$FN_{2,0}$	$FN_{2,1}$	TP_2	$FN_{2,3}$	$FN_{2,4}$	$FN_{2,NPL}$
	C_3			$FP_{3,2}$	$TN_{2,3}$		
	C_4			$FP_{4,2}$		$TN_{2,4}$	
	NTL			$FP_{NTL,2}$			$TN_{2,NTL}$

We provide (15) to (18) to calculate *TP*, *TN*, *FP*, and *FN*, respectively, for class C_k:

$$TP_k = \mathbf{M}(k,k), \quad \forall k \in \{0, \ldots, q\} \tag{15}$$

$$TN_k = \sum_{j=0}^{q}(\mathbf{M}(j,j)) - \mathbf{M}(k,k), \quad \forall k \in \{0, \ldots, q\} \tag{16}$$

$$FP_k = \sum_{j=0}^{q}(\mathbf{M}(j,k)) - \mathbf{M}(k,k), \quad \forall k \in \{0, \ldots, q\} \tag{17}$$

$$FN_k = \sum_{j=0}^{q}(\mathbf{M}(k,j)) - \mathbf{M}(k,k), \quad \forall k \in \{0, \ldots, q\} \tag{18}$$

where $\mathbf{M}$ is the MLCM and q is the number of predefined classes. Notice that the MLCM has $q + 1$ rows for q predefined classes plus one for *NTL*. Similarly MLCM has $q + 1$ columns for q predefined classes plus one for *NPL*.

D. PROPERTIES OF MLCM

For comparison purposes, we first review the properties of the multi-class confusion matrix. In multi-class, when there is exactly one label assigned to each instance (i.e., true label) and there is exactly one label predicted for each instance, the update to the confusion matrix is very simple as shown in (8) and Algorithm 1. The confusion matrix has q rows and q columns for a dataset with q classes. For each instance, there will be only one increment to one of the cells of the confusion matrix, which counts either as a *TP* to a class (also accounted as a *TN* to other classes) or as a *FN* to a class (also accounted as a *FP* to another class). Therefore the summation of all cells will be equal to the number of instances (i.e., m). The summation of each row is equal to the number of instances that belonged to the corresponding class to that row. The main diagonal of the row-wise normalized matrix shows the recall values, where the main diagonal of the column-wise normalized matrix shows the precision values. Equations (15) to (18) could be used for extracting *TP*, *TN*, *FP*, and *FN* from a multi-class confusion matrix, respectively. The summation of *FN* to all classes is equal to the summation of *FP* to all classes, since we are counting the same cells from a different perspective.

Our proposed MLCM has $q + 1$ rows and $q + 1$ columns to include the situations of *NTL* and *NPL* which are the key

points to define MLCM and covering all combinations of true and predicted labels. Since there could be more than one true label and more than one predicted label, there are situations that the incorrect prediction should be distributed over several classes. Therefore the summation of each row may be greater than the number of instances that belong to the corresponding class to that row. Consequently, the summation of all cells is greater than the total number of instances because each instance may have more than one true label. These differences have no effect on any properties of the MLCM as a confusion matrix.

For MLCM, the main diagonal of the row-wise normalized matrix shows the recall values, where the main diagonal of the column-wise normalized matrix shows the precision values. Equations (15) to (18) are used for extracting *TP*, *TN*, *FP*, and *FN* from MLCM, respectively. The summation of *FN* to all classes is equal to the summation of *FP* to all classes (e.g., see Table 7 in comparison to Table 2). Furthermore, the total *TP* in Table 7 is equal to 9, where the total *TP* in Table 2 is equal to 8 because it misses instance 3 in Table 1 as a *TP*. As discussed in section II-C, other methods for calculating *FN* and *FP* and consequently calculating metrics such as precision, recall, and F-score result in an incomplete and potentially skewed view of the classifier performance.

Another property of a multi-class confusion matrix is that each cell on the main diagonal is counted once as a *TP* and counted $q-1$ times as the *TN* to other classes, then summation of all *TN* will be $q-1$ times of summation of all *TP*. Similarly, for the MLCM, the summation of all *TN* is q times of summation of all *TP*, as there is one extra row and column (i.e., *NTL* and *NPL*) to MLCM. This property illustrates additional evidence that the current methods (e.g., `sklearn`) are incomplete as shown in Table 2 comparing to Table 7. Similar to the multi-class confusion matrix, the summation of each row of MLCM is used for calculating the weighted average of metrics such as precision, recall, or F-score (e.g., using (7)). The metrics such as precision, recall, and F-score calculated based on *TP*, *TN*, *FP*, and *FN* from MLCM will remain between zero (i.e., $TP = 0$) and one (i.e., $FP = 0$ for precision; $FN = 0$ for recall; $FP = 0$ and $FN = 0$ for F-score).

From the above comparison, we observe that MLCM has all properties that a confusion matrix has. Since there is no algorithm for creating the multi-label confusion matrix, we cannot compare the performance (complexity) of the MLCM algorithm itself to any other algorithms. For each instance, the worst-case complexity of MLCM is equal to (19)

$$\begin{aligned} W(q) &= O(Alg.2) \\ &\quad + max(O(Alg.3), O(Alg.4), O(Alg.5)) \\ &= O(q) + max(O(q), O(q^2 + q), O(q^2)) \\ &= O(q^2) \end{aligned} \tag{19}$$

where $O(Alg.i)$ is the complexity of Algorithm i and q is the number of classes. Therefore, the complexity of MLCM for the worst case is $O(mq^2)$ where m is the number of instances.

TABLE 7. One-vs-rest confusion matrix for sample instances in Table 1 using MLCM. First and second rows show [*TN, FP*] and [*FN, TP*], respectively.

(a) C_0		(b) C_1		(c) C_2		(d) *NTL*		(e) *Sum*	
4	0	7	3	8	8	8	1	27	12
6	5	4	2	0	1	2	1	12	9

IV. MLCM APPLICATION

To see the completeness and effectiveness of MLCM for processing the results of a classifier applied on a multi-label data set from a real world problem, we apply the proposed MLCM to two multi-label classifiers for data sets: i) multi-channel ECG signals, and ii) movie posters. Since the method for creating a 2-dimensional confusion matrix for multi-label classification is undefined, we compare the extracted information from MLCM such as one-vs-rest confusion matrix, precision, recall, and F1-score for each of the classes, as well as micro, macro, and weighted average of precision, recall, and F1-score of all classes to the `sklearn` library [11].

Any multi-label data set and any multi-label classifier could be used for creating the MLCM and then comparing the extracted metrics from MLCM to the metrics calculated by the method reported in [5] (i.e., implemented by [11], [12], and [13]). In the following sections IV-A and IV-B we show the results of MLCM and then collectively discuss the results in section IV-C.

A. MULTI-LABEL ECG DATA SET

We applied MLCM to a real medical waveform data set that was composed of 12-lead ECG measurement of heart signals with nine classes [23]. Each signal may be assigned with up to three labels. Almost 7% of signals were assigned to more than one class. The following performance measurements were calculated based on the results of applying a multi-label deep convolution neural network (Deep CNN) on this ECG data set.

Table 8 illustrates the raw and normalized MLCM and Table 9 shows ten one-vs-rest confusion matrices from MLCM. To compare the proposed MLCM to the current overall performance measurements, we present the `sklearn` library [11] measurements calculated using `multilabel_confusion_matrix` and `classification_report` functions in Table 10 and Table 11, respectively. Table 10 shows nine binary confusion matrices, representing a one-vs-rest confusion matrix for each of the nine classes. Table 11 shows precision, recall, and F1-score based on the results presented in Table 10. Table 12 illustrates the MLCM calculated precision, recall, and F1-score based on the results shown in Table 9.

B. MULTI-LABEL MOVIE POSTER DATA SET

We chose a data set of movie posters with eighteen classes where each poster may be assigned up to ten labels. Most of the posters were assigned to more than one class. Table 13 shows the selected rows of the normalized MLCM built on

TABLE 8. MLCM for ECG test set with nine classes.

(a) Raw MLCM

True Labels \ Predicted Labels	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	*NPL*
C_0	58	1	0	1	0	5	4	2	3	7
C_1	1	105	0	0	1	1	0	0	4	13
C_2	0	2	24	0	0	0	0	0	0	3
C_3	1	1	1	9	0	4	1	0	0	4
C_4	2	5	2	1	54	2	1	0	0	7
C_5	5	3	1	0	1	10	4	2	5	20
C_6	1	0	0	5	4	9	48	6	2	24
C_7	3	1	1	0	1	9	1	42	3	18
C_8	4	5	0	0	4	8	2	0	161	11
NTL	0	0	0	0	0	0	0	0	0	0

(b) Normalized MLCM

True Labels \ Predicted Labels	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	*NPL*
C_0	72	1	0	1	0	6	5	2	4	9
C_1	1	84	0	0	1	1	0	0	3	10
C_2	0	7	83	0	0	0	0	0	0	10
C_3	5	5	5	43	0	19	5	0	0	19
C_4	3	7	3	1	73	3	1	0	0	9
C_5	10	6	2	0	2	20	8	4	10	39
C_6	1	0	0	5	4	9	48	6	2	24
C_7	4	1	1	0	1	11	1	53	4	23
C_8	2	3	0	0	2	4	1	0	83	6
NTL	0	0	0	0	0	0	0	0	0	0

the results of applying a trained deep CNN classifier [24] on the posters validation set, where the micro-average F1-score of the classifier is 0.33. We used a very high threshold equal to 0.90 over the output of `Sigmoid` function to increase the performance of the classifier (e.g., F1-score was 0.31 for the threshold equal to 0.50) and we see bigger amounts on the *NPL* column. From the *NTL* row of this MLCM, we find that there are some movies unlabeled with any predefined genres, which was addressed by MLCM algorithm.

C. DISCUSSION

To highlight the benefits of MLCM, we will describe several scenarios comparing the extracted statistics from MLCM to current measurements tools such as `sklearn` based on the same classification model. We first show the benefits of creating MLCM followed by a discussion on extracting precise *TP*, *TN*, *FP*, and *FN* statistics from MLCM and consequently calculating precision, recall, and F-score metrics.

1) 2D CONFUSION MATRIX FOR MULTI-LABEL CLASSIFIERS

Since MLCM is the only method for creating a 2D confusion matrix for multi-label classifiers, we show the benefits of having a 2D confusion matrix for interpreting the overall performance measurements. Reviewing Table 11 (i.e., current measurement tools), we observe that class 3 (shown as C_3 in Table 8) has a low F1-score (0.51). The precision and recall for this class are 0.56 and 0.47, respectively, indicating similar *FP* and *FN* for this class. Knowing the overall F1-score only shows how the classifier performs on this specific class, whereas the precision and recall illustrate the overall *FP* and the overall *FN* related to this class, respectively. Using this report, we cannot find the distribution of *FN* over other classes and the *FP* portion related to other classes. On the other hand, by looking at MLCM, row C_3 of Table 8a or 8b, we can find that the classifier has the most confusion with class C_5 and with the no-label (*NPL*) prediction.

Returning to Table 11, we see that class 8 has a high F1-score (0.90). The precision and recall for this class are 0.90 and 0.90, respectively, which shows low *FP* and *FN* for this class. Using this report, we cannot find the distribution of *FN* over other classes and the *FP* portion of other classes to this class. On the other hand, by looking at row C_8 of Table 8a or 8b we find that the classifier has slightly equal confusion with classes C_0, C_1, C_4 to C_6, and with the no-label (*NPL*) prediction as the *FN* to class C_8. Furthermore, from column C_8 we find that a few instances belong to classes C_0, C_1, and C_5 to C_7 were classified incorrectly to class C_8 and were counted as *FP* to class C_8. For ECG data set, there is no instance with no label assigned to it, therefore all cells of the last row (*NTL*) of the MLCM are zero.

For the results of applying MLCM on Movie Posters, from Table 13 we see that classes C_7 and C_4, which represent the genres Drama and Comedy, respectively, have the most majority of *FP* predictions knowing that the majority of data were labeled with Drama and Comedy. Classes C_{14} and C_{16} (for genres Romance and Thriller, respectively) have the second most majority of *FP* predictions. In other words, we found that the classifier has *FN* prediction distributed mostly on classes C_7, C_4, C_{14}, and C_{16} while expected to predict other classes. We expect the interpretation of MLCM to shed light on data set quality and the classifier's settings. For example, the results for this experiment reveal some characteristics of this data set and the classifier, such as the data set being imbalanced and/or specific insight for setting the classifier's hyper-parameters.

2) CORRECT ACCOUNTING OF *FP*, *FN*, *TP*, *TN*

In section II-C we explained the issues related to the algorithm used by `sklearn` to create the one-vs-rest confusion matrix for multi-label classification. To show the incompleteness of its algorithm, we analyze the reported one-vs-rest confusion matrix as shown in Table 10 and compare it to the results extracted from MLCM using the equations from section III-C which were shown in Table 9. We calculated the summation of *FN* for all classes and the summation of *FP* for all classes reported by `sklearn` which were 216 and 135, respectively. We expect these two numbers to be equal, since based on the definition of *FN* and *FP* in confusion matrix, an instance of *FN* for a class is an instance of *FP* to another class and therefore their summations over all classes should be equal. This example demonstrates incorrect accounting of *FN* and *FP* by `sklearn`. On the other hand, based on the extracted statistics from MLCM, the summation of *FN* for all classes and the summation of *FP* for all classes were both equal to 243, which follows the fundamental concept of confusion matrix for multiple classes.

In another test, we divided the summation of *TN* for all classes to the summation of *TP* for all classes, which resulted in 10.45 for the `sklearn` and 9.00 for the MLCM. As we see, the result of dividing the total-TN to the total-TP using

TABLE 9. One-vs-rest confusion matrix for ECG data set extracted from MLCM. First and second rows show [TN, FP] and [FN, TP], respectively.

(a) C_0		(b) C_1		(c) C_2		(d) C_3		(e) C_4		(f) C_5		(g) C_6		(h) C_7		(i) C_8		(j) *NPL*	
453	17	406	18	487	5	502	7	457	11	501	38	463	13	469	10	350	17	511	**107**
23	58	20	105	5	24	12	9	20	54	41	10	51	48	37	42	34	161	0	0

TABLE 10. One-vs-rest confusion matrix for ECG data set using sklearn library. First and second rows show [TN, FP] and [FN, TP], respectively.

(a) C_0		(b) C_1		(c) C_2		(d) C_3		(e) C_4		(f) C_5		(g) C_6		(h) C_7		(i) C_8	
594	17	550	17	656	5	663	7	605	11	600	38	577	13	600	10	494	17
20	58	17	105	4	24	10	9	19	54	41	10	51	48	37	42	17	161

TABLE 11. scikit overall performance.

Class	precision	recall	F1-score	support
0	0.77	0.74	0.76	78
1	0.86	0.86	0.86	122
2	0.83	0.86	0.84	28
3	0.56	0.47	0.51	19
4	0.83	0.74	0.78	73
5	0.21	0.20	0.20	51
6	0.79	0.48	0.60	99
7	0.81	0.53	0.64	79
8	0.90	0.90	0.90	178
micro avg	0.79	0.70	0.74	727
macro avg	0.73	0.64	0.68	727
weighted avg	0.79	0.70	0.74	727

TABLE 12. MLCM overall performance.

Class	precision	recall	F1-score	weight
0	0.77	0.72	0.74	81
1	0.85	0.84	0.85	125
2	0.83	0.83	0.83	29
3	0.56	0.43	0.49	21
4	0.83	0.73	0.78	74
5	0.21	0.20	0.20	51
6	0.79	0.48	0.60	99
7	0.81	0.53	0.64	79
8	0.90	0.83	0.86	195
micro avg	0.68	0.68	0.68	754
macro avg	0.73	0.62	0.67	754
weighted avg	0.79	0.68	0.72	754

extracted statistics from MLCM is equal to the number of classes (q) which is what we expected, but the result of division using results from `sklearn` is meaningless.

3) ACCURATE DERIVED STATISTICS

Using MLCM *TP*, *TN*, *FP*, and *FN*, we calculated the precision, recall, and F1-score for each class, as well as micro, macro, and weighted average over all classes. Table 12 shows the calculated results incorporating all combinations of true and predicted labels for updating MLCM, providing an accurate and complete assessment of this underlying ECG classifier. Note that the cells at the last column of this table (i.e., Weight) considered *TP* and all *FN* counts to each class where based on the proposed MLCM algorithm are bigger than the number of instances that were assigned with each of the classes. We explain this property of MLCM below.

As we see, the summation of each row in Table 8a (illustrated in *weight* column of Table 12) is unmatched with the numbers in the support column of Table 11. This difference is because the method for creating Table 11 focuses on counting the instances with the true labels, which, in the case of the multi-class confusion matrix (not MLCM), will be equal to the summation of the same row of the confusion matrix. In other words, the summation of each row in the multi-class confusion matrix is equal to the number of instances in the corresponding class, but the summation of each row in MLCM is rarely equal and mostly greater than the number of instances in the corresponding class. This is the consequence of increasing MLCM cells for both situations of under and over prediction where both counted as *FN* to the corresponding class, which is ignored in the one-vs-rest confusion matrix created by `sklearn` library. We explained this issue in section II-C.

The MLCM algorithm may count more than one *FN*/*FP* for each instance of a multi-label data set for incorrect prediction(s) while there is only one *FN*/*FP* for incorrect prediction of each instance of a multi-class data set. Counting more than one *FN*/*FP* is a correct way to consider the combination of true and predicted labels where in a multi-label data set each instance may be assigned with more than one label, which could be interpreted as more than one instance. Applying the MLCM algorithm on multi-class data sets returns exactly the same confusion matrix that the multi-class confusion matrix algorithm produces.

4) IMPROVED MODELLING

Using the proposed MLCM, the modeler of a multi-label classifier would be able to concisely see the distribution of *FN* predictions over each class, quickly finding the situations that a classifier has the most confusion. Knowing this information, the modeler can focus on the classes that have the most confusion for the classifier and address the problem by examining the source data, extracted features, or the configuration of the classifier. In addition, MLCM could be used for creating a weight matrix for the cost-sensitive loss metrics [10], especially in the case of imbalanced data sets.

TABLE 13. Selected rows of normalized MLCM for movie posters classification.

		Predicted Labels																		
	Classes	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	*NPL*
True Labels	C_0	6	1	1	1	15	3	4	39	1	2	0	3	0	1	8	1	5	0	7
	C_4	4	3	1	1	19	4	3	34	2	2	1	3	0	2	6	2	7	0	7
	C_7	1	2	1	0	7	2	1	66	1	2	0	2	0	1	5	1	5	0	0
	C_{14}	6	5	1	1	10	7	4	22	1	3	0	4	0	2	8	2	8	0	17
	C_{16}	3	3	1	1	18	3	3	34	1	2	0	3	0	1	8	1	8	0	8
	NTL	3	1	1	1	18	2	3	50	1	1	0	3	0	1	8	2	4	0	0

TABLE 14. Summary of benefits from proposed MLCM.

Metrics	Advantage of using MLCM
MLCM matrix	Concise statistics, helps finding to what classes, the most overlap of false prediction occurred
FN, *FP*, *TP*, *TN*	Extracts precise statistics from MLCM by considering all combinations of true and predicted labels
Precision, Recall, and F-score	Calculated based on extracted statistics from MLCM, provides precise overall statistics for each class
micro, macro, and weighted avg	Calculates precise micro, macro, and weighted average of Precision, Recall, and F-score for all classes

V. CONCLUSION

A clear understanding of the correct and incorrect distribution of the classifier's prediction is imperative for assessing and refining a classifier. A 2D confusion matrix could give a clear distribution of the classifier's prediction in one view, which is currently undefined for the multi-label data sets. Our contribution is to define a method for creating a multi-label confusion matrix.

Furthermore, by extracting the accurate *TP*, *TN*, *FP*, and *FN* information of each class for multi-label classifiers from MLCM, we can calculate the precise overall statistics such as precision, recall, and F-score using the same equations used for multi-class classifiers with normalized results. This is in contrast to the current measurement methods that return ambiguous statistics while focusing on one label at a time, ignoring the combination of true and predicted labels together. The result of ignoring these combinations produces inflated precision, recall, and F-score. MLCM calculates accurate precision, recall, and F-score. Table 14 summarizes the benefits of our contribution by proposing the MLCM method.

Due to the possibility of not predicting any labels for some, or all, of the true labels, we propose to add the no predicted label case (i.e., extra column to the confusion matrix NPL), which is the key for creating an accurate multi-label confusion matrix. Similarly, there is a possibility of not assigning any label for some instances, thus we propose to add the no true label case (i.e., extra row to the confusion matrix NTL), which is the key to the generalization of creating a multi-label confusion matrix.

Trust and explainability are other important considerations where development tools play an important role. Consider the scenario where the user of a classifier wants to know the level of trust for the classifier in, for example, a disease classification. One way of measuring trust and/or explainability is to quantify the overlap of incorrect predictions of a class with classes of the same or different category (e.g., based on the cause and/or treatment for the labeled disease), which now is possible using MLCM.

While our proposed method counts *FN* by distributing each of incorrectly predicted labels over all of not predicted labels (or over all of the correctly predicted classes in the case that all true labels are predicted correctly), investigating other possible methods for counting *FN* is a suggestion for the future work.

REFERENCES

[1] B. Altınel and M. C. Ganiz, "Semantic text classification: A survey of past and recent advances," *Inf. Process. Manage.*, vol. 54, no. 6, pp. 1129–1153, Nov. 2018.
[2] Z. Li, Y. Fan, B. Jiang, T. Lei, and W. Liu, "A survey on sentiment analysis and opinion mining for social multimedia," *Multimedia Tools Appl.*, vol. 78, no. 6, pp. 6939–6967, 2019.
[3] M. Fatima and M. Pasha, "Survey of machine learning algorithms for disease diagnostic," *J. Intell. Learn. Syst. Appl.*, vol. 9, no. 1, pp. 1–16, 2017.
[4] J. L. Martinez-Rodriguez, A. Hogan, and I. Lopez-Arevalo, "Information extraction meets the semantic web: A survey," *Semantic Web*, vol. 11, no. 2, pp. 255–335, Feb. 2020.
[5] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1819–1837, Jun. 2014.
[6] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Inf. Process. Manage.*, vol. 45, no. 4, pp. 427–437, Jul. 2009.
[7] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," in *Data Mining and Knowledge Discovery Handbook*, 2nd ed., O. Maimon and L. Rokach, Eds. Cham, Switzerland: Springer, 2010, ch. 34, pp. 667–685.
[8] R. B. Pereira, A. Plastino, B. Zadrozny, and L. H. Merschmann, "Correlation analysis of performance measures for multi-label classification," *Inf. Process. Manage.*, vol. 54, no. 3, pp. 359–369, May 2018.
[9] N. Spolaor, E. A. Cherman, J. Metz, and M. C. Monard, "A systematic review on experimental multi-label learning," Inst. Math. Comput. Sci., Chennai, India, Tech. Rep. ISSN-0103-2569, Feb. 2013.
[10] C. Elkan, "The foundations of cost-sensitive learning," in *Proc. 17th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2001, pp. 973–978.
[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, and B. Thirion, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12 no. 10, pp. 2825–2830, 2012.
[12] G. Tsoumakas, E. Spyromitros-Xioufis, J. Vilcek, and I. Vlahavas, "MULAN: A Java library for multi-label learning," *J. Mach. Learn. Res.*, vol. 12, pp. 2411–2414, Jul. 2011.
[13] J. Read, P. Reutemann, B. Pfahringer, and G. Holmes, "MEKA: A multi-label/multi-target extension to Weka," *J. Mach. Learn. Res.*, vol. 17, no. 21, pp. 1–5, 2016.
[14] K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier, "On label dependence and loss minimization in multi-label classification," *Mach. Learn.*, vol. 88, no. 1, pp. 5–45, Jul. 2012.
[15] W. Waegeman, K. Dembczynski, A. Jachnik, W. Cheng, and E. Hüllermeier, "On the Bayes-optimality of f-measure maximizers," *J. Mach. Learn. Res.*, vol. 15, pp. 3513–3568, 2014.
[16] O. Koyejo, N. Natarajan, P. Ravikumar, and I. S. Dhillon, "Consistent multilabel classification," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 3321–3329.

[17] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Džeroski, "An extensive experimental comparison of methods for multi-label learning," *Pattern Recognit.*, vol. 45, no. 9, pp. 3084–3104, Sep. 2012.
[18] J. Lee, H. Kim, N.-R. Kim, and J.-H. Lee, "An approach for multi-label classification by directed acyclic graph with label correlation maximization," *Inf. Sci.*, vol. 351, pp. 101–114, Jul. 2016.
[19] X.-Z. Wu and Z.-H. Zhou, "A unified view of multi-label performance measures," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 5778–5791.
[20] T. Eftimov and D. Kocev, "Performance measures fusion for experimental comparison of methods for multi-label classification," in *Proc. AAAI, Spring Symp. Combining Mach. Learn. With Knowl. Eng. (AAAI-MAKE)*, 2019, pp. 1–10.
[21] M. Bogaert, J. Lootens, D. Van den Poel, and M. Ballings, "Evaluating multi-label classifiers and recommender systems in the financial service sector," *Eur. J. Oper. Res.*, vol. 279, no. 2, pp. 620–634, Dec. 2019.
[22] A. K. Menon, A. S. Rawat, S. J. Reddi, and S. Kumar, "Multilabel reductions: what is my loss optimising," in *Proc. 33rd Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2019, pp. 10600–10611.
[23] E. A. Perez Alday, A. Gu, A. J Shah, C. Robichaux, A.-K. Ian Wong, C. Liu, F. Liu, A. Bahrami Rad, A. Elola, S. Seyedi, Q. Li, A. Sharma, G. D. Clifford, and M. A. Reyna, "Classification of 12-lead ECGs: The PhysioNet/Computing in cardiology challenge 2020," *Physiolog. Meas.*, vol. 41, no. 12, Dec. 2020, Art. no. 124003.
[24] A. Maiza. (Oct. 2020). *Multi-Label Image Classification in Tensorflow 2.0*. [Online]. Available: https://towardsdatascience.com/multi-label-image-classification-in-tens%orflow-2-0-7d4cf8a4bc72

MOHAMMADREZA HEYDARIAN received the B.S. degree in computer engineering from Shahid Beheshti University, Tehran, Iran, the M.S. degree in machine intelligence and robotics from Shiraz University, Shiraz, Iran, and the Ph.D. degree in computer science from McMaster University, Hamilton, ON, Canada.

He was a Postdoctoral Fellow with the Imaging Research Laboratories, Robarts Research Institute, London, ON, Canada, and an Assistant Professor with Vali-e-Asr University, Rafsanjan, Iran. Since 2020, he has been a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, McMaster University. His research interests include image and signal processing, machine learning, evolutionary algorithms, and trust in AI.

Dr. Heydarian was awarded the Michael G. DeGroote Health Innovation, Commercialization and Entrepreneurship Fellowship from the Department of Medicine, McMaster University.

THOMAS E. DOYLE (Senior Member, IEEE) received the Ph.D. degree from the University of Western Ontario. He is currently an Associate Professor with the Department of Electrical and Computer Engineering and a Member of the School of Biomedical Engineering, Faculty of Engineering, McMaster University. In addition, he is a Faculty Affiliate with the Vector Institute for Artificial Intelligence. From 2014 to 2019, he was the Director of the McMaster eHealth Graduate Program. His research interests include artificial intelligence and machine learning for human health and performance, human-AI partnership, and trust in autonomous medical advisory systems. His current and past research awards are received from CIHR, NSBRI, SOSCIP, MITACS, HHS, and Canadian DND IDEaS.

REZA SAMAVI (Senior Member, IEEE) received the Ph.D. degree from the University of Toronto. He is currently an Assistant Professor with the Department of Electrical, Computer, and Biomedical Engineering, Faculty of Engineering, Ryerson University, and a Faculty Affiliate with the Vector Institute for Artificial Intelligence. His research interests include trustworthy machine learning and data security and privacy. For his research on AI and security, he has received several research grants from NSERC, SOSCIP, MITACS, HHS, and IDEaS. For his research on information privacy, he has received the Privacy Technologies Research Award from IBM and the Privacy by Design Research Award from the Information and Privacy Commissioner of Ontario.