# PLM: Partial Label Masking for Imbalanced Multi-label Classification Supplementary Material

Kevin Duarte
kevin_duarte@knights.ucf.edu

Yogesh Rawat
yogesh@crcv.ucf.edu

Mubarak Shah
shah@crcv.ucf.edu

Center for Research in Computer Vision
University of Central Florida, Orlando, Florida, USA

In the supplementary material we include a description of different multi-label evaluation metrics used (Section 1), a more detailed description of how the discrete output distributions are computed (Section 2), and additional details on the imbalanced MultiMNIST dataset (Section 3). Furthermore, we include additional results on the MSCOCO dataset (Section 4).

## 1. Metrics

In this section, we explain the metrics used in our multi-label classification experiments. In an evaluation set, there are $N$ images with corresponding labels $\{\mathbf{y}^{(1)}, ..., \mathbf{y}^{(N)}\}$ where $\mathbf{y}^{(i)} = [y_1^{(i)}, ..., y_C^{(i)}]$ is the $i^{th}$ binary label vector; note that multiple elements in this label vector may be non-zero in multi-label classification. For each sample, the classifier predicts class probabilities to which a threshold (0.5 in our experiments) is applied to obtain binary prediction vectors $\{\hat{\mathbf{y}}^{(1)}, ..., \hat{\mathbf{y}}^{(N)}\}$.

**Per-class Precision and Recall** The precision and recall of a classifier on a single class $c$ is given by

$$\begin{aligned} \mathrm{P}(c) &= \frac{\sum_{i=1}^{N} \mathbb{1}\left[y_c^{(i)} = \hat{y}_c^{(i)} = 1\right]}{\sum_{i=1}^{N} \mathbb{1}\left[\hat{y}_c^{(i)} = 1\right]}, \\ \mathrm{R}(c) &= \frac{\sum_{i=1}^{N} \mathbb{1}\left[y_c^{(i)} = \hat{y}_c^{(i)} = 1\right]}{\sum_{i=1}^{N} \mathbb{1}\left[y_c^{(i)} = 1\right]}, \end{aligned} \tag{1}$$

respectively, where $\mathbb{1}[...]$ is the indicator function. To obtain the reported metric, we average the precision (or recall) over all classes:

$$\begin{aligned} \text{Precision} &= \frac{1}{C}\sum_{c=1}^{C} \mathrm{P}(c), \\ \text{Recall} &= \frac{1}{C}\sum_{c=1}^{C} \mathrm{R}(c). \end{aligned} \tag{2}$$

In these metrics, each class is treated equally regardless of the number of samples in the test set.

**F1-score** The F1-score is the harmonic mean between precision and recall:

$$\mathrm{F}_1(c) = \frac{2\mathrm{P}(c)\,\mathrm{R}(c)}{\mathrm{P}(c) + \mathrm{R}(c)}. \tag{3}$$

The final F1-score is averaged over all classes:

$$\text{F1-score} = \frac{1}{C}\sum_{c=1}^{C} \mathrm{F}_1(c). \tag{4}$$

**0-1 Accuracy** This metric measures how often the network is able to output predictions which exactly match the ground-truth labels. It is computed as follows:

$$\text{0-1 Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left[\mathbf{y}^{(i)} = \hat{\mathbf{y}}^{(i)}\right]. \tag{5}$$

Since this metric is averaged over all samples, as opposed to all classes, it tends to be biased towards more frequent classes. Therefore, this metric tends to be a poor measure of model performance if there is imbalance present in the test set.

## 2. Output Distributions

In this section, we describe in further detail how the discrete distributions $P_c^+$, $\hat{P}_c^+$, $P_c^-$, and $\hat{P}_c^-$ are formed. Given the set of probabilities described in equation 4, we form these discrete distributions through a binning operation. Since the output probabilities are within the range $[0, 1]$, we create $\tau$ bins, each of width $1/\tau$. The probabilities are placed within these bins to obtain a histogram, which is then normalized to obtain a discrete probability distribution.

We present a toy example to describe the process. Suppose there is a single-class dataset with 10 samples and following labels: $\{1, 1, 1, 1, 1, 0, 0, 0, 0, 0\}$.

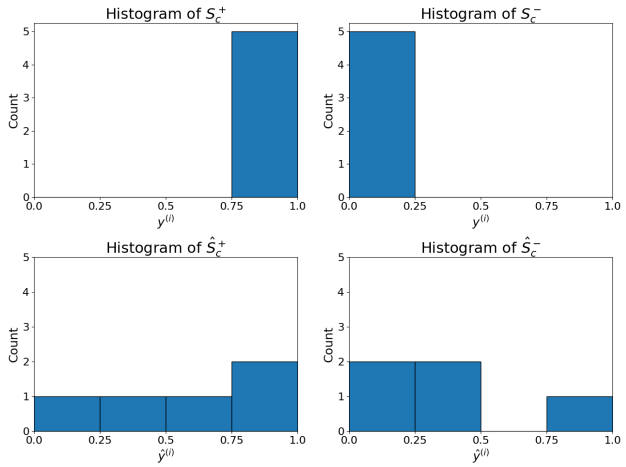


Figure 1. The histograms constructed from the sets of ground-truth labels and predicted probabilities for the toy example described in section 2.

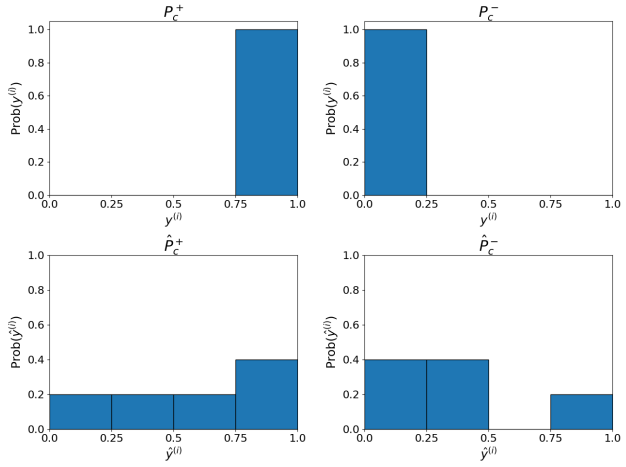


Figure 2. The discrete probability distributions constructed from the sets of ground-truth labels and predicted probabilities for the toy example described in section 2.

For these samples, the network predicts probabilities: $\{0.2, 0.6, 0.95, 0.99, 0.45, 0.1, 0.15, 0.8, 0.4, 0.3\}$. The ground-truth and predicted output sets are as follows: $S_c^+ = \{1, 1, 1, 1, 1\}$, $S_c^- = \{0, 0, 0, 0, 0\}$, $\hat{S}_c^+ = \{0.2, 0.6, 0.95, 0.99, 0.45\}$, and $\hat{S}_c^- = \{0.1, 0.15, 0.8, 0.4, 0.3\}$. From these sets we can construct the histogram; the histograms (with $\tau = 4$) and their corresponding discrete distributions are shown in figures 1 and 2 respectively.

In general, these distributions tend to be shifted towards the left (i.e. probability outputs closer to zero) for classes with fewer samples and shifted towards the right for classes with many samples. This behaviour can be seen in figure 3, which contains the distributions produced by a classifier when trained on the MultiMNIST dataset for 10 epochs. PLM makes use of this observation to mask individual labels such that predicted probability distributions more resemble to ground-truth distribution.

| Digit | Train | Test |
|---|---|---|
| 0 | 9485 | 17840 |
| 1 | 6167 | 19080 |
| 2 | 3967 | 18256 |
| 3 | 2476 | 18080 |
| 4 | 1508 | 17856 |
| 5 | 875 | 17136 |
| 6 | 538 | 17664 |
| 7 | 331 | 18224 |
| 8 | 164 | 17792 |
| 9 | 105 | 18072 |
| Total | 14694 | 90000 |

Table 1. The number of samples for each digit in the Imbalanced MultiMNIST dataset.

## 3. Imbalanced MultiMNIST Dataset

**Dataset** We construct the Imbalanced MultiMNIST dataset by superimposing two MNIST [2] digits into a single image. We begin by sampling the MNIST training set similarly to how [1] create the Imbalanced CIFAR dataset with an imbalance of $\rho = 100$. These images are padded to become $32 \times 32$ and shifted randomly by $-6$ to 6 pixels, both vertically and horizontally. Then, each image is randomly paired with another digit in the training set, which is also padded and shifted randomly. These two digits are superimposed upon each other to create a $32 \times 32$ image with two digits. Due to cooccurrence of the same digit, within a sample, the final imbalance of the dataset becomes 90.33.

For evaluation, we select all samples in the MNIST test set and perform the same padding and shifting operations. For each digit, we select other random test digits that have different labels, which are superimposed to make 9 samples with two digits each. This results in 90000 test samples. The test set is roughly balanced, with all classes having a similar number of samples. The number of samples for each class can be found in Table 1.

## 4. Additional Results

**Additional MSCOCO Results** We include more results on the MSCOCO dataset. Figure 4 contains the class-wise precision and recall scores on the MSCOCO dataset. We find that using PLM leads to an large improvement in recall for the tail classes, and a slight improvement in precision for the head classes (7.86% improvement on the 10 most frequent classes). This is in line with the observation that PLM reduces over-prediction on the head classes while reducing under-prediction on the tail classes (as well as some

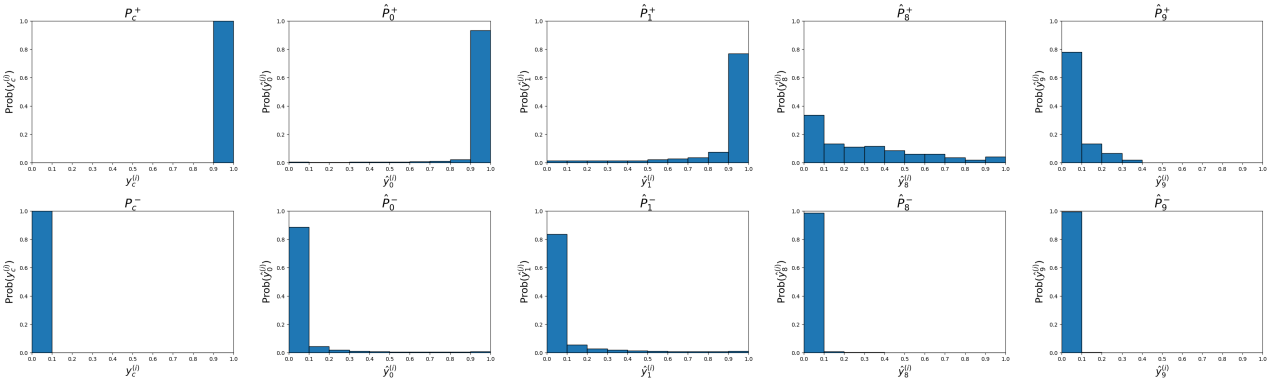

Figure 3. The discrete probability distributions ($\tau = 10$) obtained from the MultiMNIST training set. The first column consists of the ground-truth distributions. The distributions on the top are for the positive labels, and the distributions on the bottom are for the negative labels. Distributions for the head classes (columns 2 and 3) tend to be skewed more towards the right (i.e. predictions closer to 1); distributions for the tail classes (columns 4 and 5) tend to be skewed more towards the left (i.e. predictions closer to 0).

difficult classes).

## References


[1] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9268–9277, 2019. 2

[2] Yann LeCun, Corinna Cortes, and Chris Burges. Mnist handwritten digit database, 1998. 1998. 2

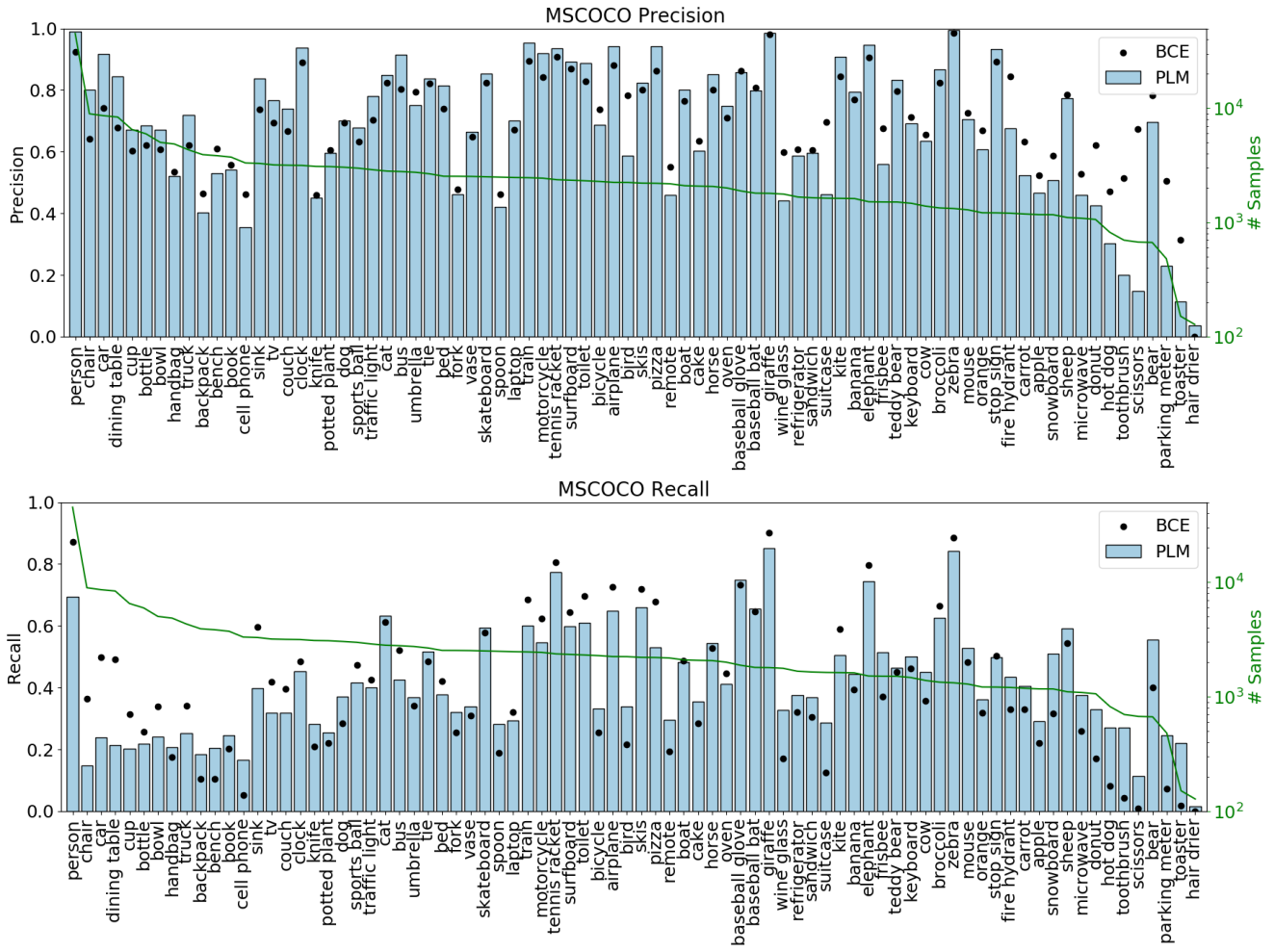


Figure 4. The class-wise precision and recall scores on the MSCOCO dataset. The classes are ordered based on the number of samples in the training set. The bars represent the results achieved by using the PLM method, and the points are the results when using BCE.