Have the Cake and Eat It Too? Higher Accuracy and Less Expense when Using Multi-label ML APIs Online

Anonymous Authors[1]

Abstract

The fast growing machine learning as a service industry has incubated many APIs for multi-label classification tasks such as OCR and multi-object recognition. The heterogeneity in those APIs' price and performance, however, often forces users to make a choice between accuracy and expense. In this work, we propose FrugalMCT, a principled framework that jointly maximizes the accuracy while minimizes the expense by adaptively selecting the APIs to use for different data.. FrugalMCT combines different APIs' predictions to improve accuracy, and selects which combination to use to respect expense constraint. Preliminary experiments using ML APIs from Google, Microsoft, and other providers for multi-label image classification show that FrugalMCT often achieves more than 50% cost reduction while matching the accuracy of the best single API.

1. Introduction

The growing machine learning as a service (MLaaS) industry has incubated plenty of commercial ML APIs. A major focus of those APIs is multi-label classification. For example, one can use Google's prediction API (Goo) to tag an image with all possible keywords for \$0.0015, or Microsoft's API (Mic) for \$0.010. Besides price, the accuracy performance of those APIs is also diverse on different datasets. Users often pay for high expense to reach high accuracy, or tolerate limited performance to ensure low cost.

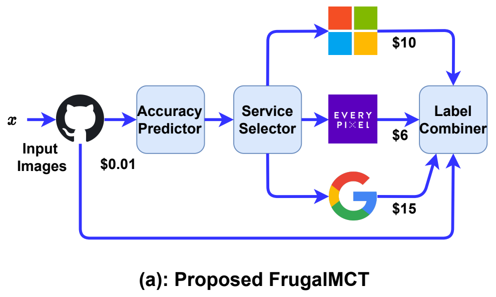
Jointly optimizing accuracy and cost was studied for single-label ML APIs. For example, FrugalML (Chen et al., 2020) uses a decision rule for each possible label to determine which API to call, and involves a non-convex optimization problem with computational complexity exponential in the number of distinct labels. Such a high complexity prevents it from being used for tasks with large number of labels, such as multi-label classification. Furthermore, FrugalML ignores correlation between different APIs' predictions, potentially leading to limited accuracy. Thus, this paper aims to solve these significant limitations and address the question: *how to design efficient multi-label ML API selection strategies to jointly optimize accuracy as well as expense?*

Contribution. Towards addressing this question, we propose FrugalMCT, a principled framework that learns the strengths and weaknesses of different combinations of available APIs for multi-label classification tasks, and efficiently selects the optimal combinations of APIs to call for different data items and budget constraints. As shown in Fig. 1 (a), FrugalMCT consists of three main components: an accuracy predictor, a service selector, and a label combiner. The accuracy predictor estimates the accuracy of each API combination on a particular input based on the features and predicted labels of that input. Then a fast service selector is invoked to determine which combination to use for accuracy and budget balance. Finally, the label combiner uses the called APIs' prediction to obtain a more accurate outcome. Fig. 1 (b) gives an example. We first invoke a GitHub model which returns *person* and *tennis racket*. Accuracy predictor believes that combining it with the Everypixel gives a much higher accuracy. Thus, we invoke Everypixel and combine their prediction to obtain *{person, sports ball, tennis racket}*. Note that this prediction is not possible with any single API.

In fact, preliminary empirical study demonstrates that FrugalMCT can significantly improve the accuracy and reduce the cost. Quantitatively, across experiments using real world APIs from Google, Microsoft, and Everypixel, we observe that FrugalMCT typically leads to over 50% (as high as 86%) cost reduction when matching the best commercial API's performance. Moreover, when using the same cost as the best commercial API, FrugalMCT can improve the performance by up to 8%.

2. Preliminaries and Related Work

Multi-label classification Tasks. In multi-label classification tasks, the goal is to assign a label set $Y \subseteq \mathcal{Y}$ to any data point $x \in \mathcal{X}$. In contrast to basic supervised learning, in multi-label learning each data point is associated with a set of labels instead of a single label. Many real world ML APIs aim at such tasks. Consider, e.g., image tagging, where $\mathcal{X}$ is a set of images and $\mathcal{Y}$ is the set of all tags. Example label sets could be $\{$*person*, *car*$\}$ or $\{$*bag*, *train*, *sky*$\}$.



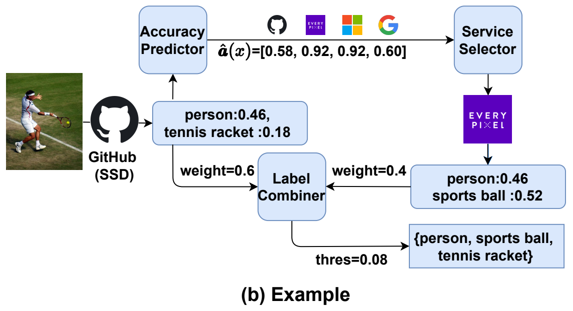

Accuracy Predictor
$\hat{a}(x)$=[0.58, 0.92, 0.92, 0.60]
Service Selector
GitHub (SSD)
person:0.46, tennis racket :0.18
weight=0.6
Label Combiner
weight=0.4
person:0.46 sports ball :0.52
thres=0.08
{person, sports ball, tennis racket}
(b) Example

Figure 1. Demonstration of FrugalMCT. **(a)** FrugalMCT workflow. **(b)** Example of FrugalMCT's behavior using real world APIs, where the prediction of GitHub and Everypixel APIs is combined as FrugalMCT estimates that their combination has a much higher estimated accuracy than using GitHub API alone.

MLaaS. With the growing importance of MLaaS APIs (Goo; Mic), most research has largely focused on evaluating individual API's performance (Yao et al., 2017; Koenecke et al., 2020). Recent work (Chen et al., 2020) studies API calling strategies for single label tasks. To our knowledge, FrugalMCT is the first system for optimizing calling strategies for multi-label prediction APIs. Here we consider a MLaaS market consisting of K multi-label ML APIs. For a data point x, the kth service returns to the user a set of labels, and their quality scores, denoted by $Y_k(x) \subseteq \mathcal{Y} \times [0,1]$. $\boldsymbol{c} \in \mathbb{R}^K$ denotes the unit cost of all APIs.

Ensembles for Multi-Label Classifications: Ensemble learning is a natural approach to combine different predictors' output. Several ensemble methods have been developed, such as using pruned sets (Read et al., 2008), and random subsets (Tsoumakas & Vlahavas, 2007). Almost all of those ensemble methods require training of the base classifiers, but the ML APIs are black box to the users. Also, while standard ensemble methods focus on exclusively improving accuracy, FrugalMCT explicitly considers cost of each API and enforces a budget constraint.

3. FrugalMCT Framework

Now we introduce FrugalMCT, a principled framework to adaptively select ML APIs for multi-label classification tasks given a budget. We generalize the scheme in Figure 1 (a) to K ML services. As shown in Figure 2, FrugalMCT contains three main components: an accuracy estimator, a service selector, and a label combiner. Given a data point x, it first calls some base service, denoted by *base*, which is one of the K APIs, and obtains $Y_{base}(x)$. Next, an accuracy predictor produces a vector $\hat{\boldsymbol{a}}(x) \in [0,1]^K$, whose kth value estimates the accuracy of the label set produced by the label combiner using base's and kth API's outputs. The service selector $s(\cdot) : \mathcal{X} \mapsto [K]$ then decides if and which *add-on service* needs to be invoked. Finally, a label combiner generates a label set by combining the predictions from the base and add-on APIs. In the following, we explain the key components in more detail.

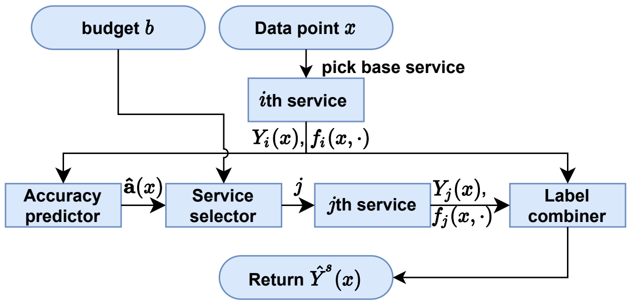


Figure 2. Overview of FrugalMCT. Given a data point, FrugalMCT first invokes a base service. Based on its output, an accuracy predictor estimates the performance of different APIs. Next, an add-on service is selected based on the predicted accuracy and budget. Finally, the add-on and base services' predictions are combined to return FrugalMCT's prediction.

Accuracy Predictor. The accuracy predictor $\hat{\boldsymbol{a}}(\cdot)$ is created via two steps. First, we generate a feature vector for every data point in the training dataset $\mathbb{X}^{Tr} \triangleq \{x_1^{Tr}, x_2^{Tr}, \cdots, x_{N^{Tr}}^{Tr}\}$. Generally the feature vector can be any embedding of the data point x and base service prediction $Y_{base}(x)$. Here we adopt a simple approach: Given the label set, a $|\mathcal{Y}|$ dimensional vector is generated using one hot encoding on $Y_{base}(x)$. For example, given $\mathcal{Y} = \{person, car, bike\}$ and $Y_{base}(x) = \{(person, 0.8), (car, 0.7)\}$, the generated feature vector is $[0.8, 0.7, 0]$. The next step is to train the accuracy predictor. For each $x_n^{Tr} \in \mathbb{X}^{Tr}$, as its true label sets and prediction from each API are available, we can construct its true accuracy vector $\boldsymbol{a}(x_n^{Tr}) \in [0,1]^K$, whose kth element is the accuracy of the label produced by the label combiner using base and kth service predictions. Then we can train some regressor (e.g., random forest) to map the feature vector to the accuracy vector. We use standard multi-label accuracy[1] (Zhang & Zhou, 2014) as a concrete metric. FrugalMCT can just as easily use another metric such as F1-score, precision or subset accuracy.

[1] $\frac{\|Y \cap Y'\|}{\|Y \cup Y'\|}$ where Y/Y' is the true/predicted label set.

An Online Service Selector. A key part of FrugalMCT is to design the service selector s: given a budget b and the estimated accuracy $\hat{\boldsymbol{a}}(x)$, which service should be invoked? Let $\mathbb{X} \triangleq \{x_1, x_2, \cdots, x_N\}$ be the entire unlabeled dataset to be classified, and $S \triangleq \{1, 2, \cdots, K\}^{\mathbb{X}}$ be the set of all possible functions mapping each data point in $\mathbb{X}$ to an API. For any $s \in S$, $s(x) = \textit{base}$ implies no add-on API is needed, and $s(x) = k \neq \textit{base}$ implies kth API is invoked. Our goal is to find some mapping $s \in S$ to maximize the estimated accuracy while satisfies the budget constraint, formally stated as below.

Definition 1. *Let $\boldsymbol{Z}^*_{n,k}$ be the optimal solution to the budget aware API selection problem*

$$\begin{aligned} \max_{\boldsymbol{Z}\in\mathbb{R}^{N\times K}:} & \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{Z}_{n,k}\hat{\boldsymbol{a}}_k(x_n) \\ s.t. & \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1,k\neq base}^{K} \boldsymbol{Z}_{n,k}\boldsymbol{c}_k + \boldsymbol{c}_{base} \leq b \qquad (3.1) \\ & \sum_{k=1}^{K} \boldsymbol{Z}_{n,k} = 1, \boldsymbol{Z}_{n,k} \in \{0,1\}, \forall n, k \end{aligned}$$

Then the optimal strategy is $s^*(x_n) \triangleq \arg\max_k \boldsymbol{Z}^*_{n,k}$.

The objective quantifies the average accuracy, the first constraint models the budget requirement, and the last two enforce only one add-on API is picked for each data point.

Two challenges exist for solving Problem 3.1. First is computational cost: it is an integer linear program and generally NP-hard. Second, the input data x_n (and the accuracy vector $\hat{\boldsymbol{a}}(x_n)$) comes sequentially, and the API needs to be selected before observing the future data.

To tackle this challenge, we propose an efficient online approach, which requires $O(K)$ computations per round. The key idea is to explicitly balance between accuracy and cost at every iteration. More precisely, for a given data point x_n and $p \in \mathbb{R}$, let us define a strategy

$$s^p(x_n) = \begin{cases} \arg\max_k \hat{\boldsymbol{a}}_k(x_n) - p\boldsymbol{c}_k \mathbb{1}_{k\neq base} & \text{budget left} \\ base & o/w \end{cases}$$

Here, the second case ensures the budget requirement. p is a parameter to balance between accuracy $\hat{\boldsymbol{a}}(x_n)$ and cost $\boldsymbol{c}$. When $p = 0$, $s^p(x_n)$ selects the API with highest estimated accuracy. When p is large enough, $s^p(x_n)$ enforces to pick the base API. In fact, larger value of p implies more weights on cost and smaller p favors more the accuracy.

The performance of this strategy depends on the choice of p. Ideally, we want to select the smallest p such that the budget constraint is satisfied (i.e., the second case in $s^p(x_n)$is not invoked). This can be easily achieved via a binary search of p if the entire dataset is available. In an online setting, we can obtain the best p on the training dataset $\mathbb{X}^{Tr}$ first, and then increase it slightly (say, by 1%) to ensure it works on the incoming data stream with high chance.

Table 1. ML services used for each task. Price unit: USD/10,000 queries. The GitHub model is a single shot detector (SSD) (SSD) pretrained on Open Images V4 (Kuznetsova et al., 2020).

ML Service	Price	ML Service	Price
SSD (SSD)	<0.01	Everypixel (Eve)	6
Microsoft (Mic)	10	Google (Goo)	15

Label Combiner. The label combiner contains two phases. First, a new label set associated with its quality function is produced. The label set is simply the union of that from the base service and add-on service. The quality score is a weighted sum of the score from both APIs, controlled by w. For example, suppose the base predicts $\{(\textit{person}, 0.8), (\textit{car}, 0.7)\}$ and the add-on predicts $\{(\textit{car}, 0.5), (\textit{bike}, 0.4)\}$. Given $w = 0.3$, new confidence for *person* is $0.3 \times 0.8 = 0.24$, for *car* is $0.3 \times 0.7 + 0.7 \times 0.5 = 0.46$, and for *bike* is $0.7 \times 0.4 = 0.28$. Thus the combined set is $\{(\textit{person}, 0.24), (\textit{car}, 0.46), (\textit{bike}, 0.28)\}$. Next, a threshold θ is applied to remove labels with low confidence. The parameters w and θ are global hyperparameters for each dataset, and can be obtained by an efficient searching algorithm to maximize the overall performance.

4. Experiments

We compare the accuracy and incurred costs of FrugalMCT to that of real world ML services on a few multi-label datasets to show how much FrugalMCT can improve the accuracy and reduce cost.

Tasks, ML services, and Datasets. The preliminary experiments focuses on multi-label image classification, i.e., obtaining all keywords associated with an image. We use three commercial APIs from Google. Microsoft, and Everypixel, plus an open source GitHub model, as summarized in Table 1. Table 2 lists the statistics for all the datasets.

Table 2. Dataset Statistics.

Dataset	Size	# Labels	Dist Labels
PASCAL	11540	16682	20
MIR	25000	92909	24
COCO	123287	357662	80

Accuracy Predictors. We use a random forest regressor as the accuracy predictor for all the datasets. The accuracy predictor is trained on half of the datasets.

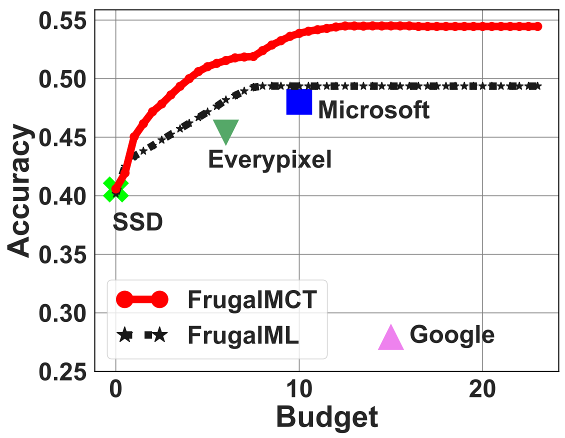


Figure 3. Accuracy cost trade-offs on COCO. Compared to FrugalML (Chen et al., 2020) or any single API, FrugalMCT significantly improves the accuracy given any budget target.

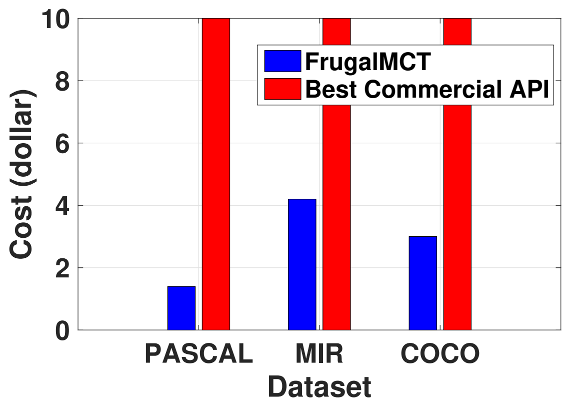


Figure 4. Cost savings by FrugalMCT. To reach the same accuracy of the best commercial APIs, FrugalMCT often leads to more than 50% (up to 86%) cost reduction. .

Accuracy and Cost Trade-offs. We first study the accuracy and cost trade-offs achieved by FrugalMCT on the dataset COCO, shown in Figure 3. We first note that, perhaps surprisingly, expensive APIs may not always have the best performance. In fact, Google API is the most expensive, but its accuracy is worse than the cheap GitHub API. Compared to any single API, FrugalMCT, This shows the importance of carefully selecting multi-label APIs even with a large budget. Compared to any single API, FrugalMCT allows users to pick any point in its trade-off curve and offers substantial more flexibility. In addition, FrugalMCT often achieves higher accuracy than any ML services it calls. Compared to the best commercial API (Microsoft), FrugalMCT gives a 4% accuracy gain without extra cost. FrugalMCT also outperforms FrugalML with the same budget.

Cost Savings. Next, we evaluate how much cost can be saved by FrugalMCT to reach the highest accuracy produced by a single API on different tasks, shown in Figure 4. Note that FrugalMCT can typically save more than 50% of the cost. This is probably because (i) the accuracy estimator enables the API selector to identify when the base service's prediction is reliable and to avoid unnecessarily calling add-on services, and (ii) when add-on API is invoked, the apt combination of the base and add-on services leads to a significant accuracy improvement.

5. Conclusion

In this work, we propose FrugalMCT, a generic framework to adaptively select from multi-label ML APIs to jointly optimize accuracy performance and budget requirements. How to efficiently use multi-label APIs is important due to its wide applications in practice, although there is limited study in the ML literature. Preliminary empirical evaluation using real APIs shows significant cost reduction and accuracy improvements by FrugalMCT. As a next step, We are working on in-depth theoretical analysis for FrugalMCT as well as more comprehensive empirical study. To stimulate more research on MLaaS, we also plan to release the datasets used to develop FrugalMCT.

References

Everypixel Image Tagging API. https://labs.everypixel.com/api. [Accessed Oct-2020].

Google Vision API. https://cloud.google.com/vision. [Accessed Oct-2020].

Microsoft computer vision API. https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision. [Accessed Oct-2020].

SSD, a multi-label image tag tool from GitHub/TensorflowHub. https://tfhub.dev/google/openimages_v4/ssd/mobilenet_v2/1. [Accessed Oct-2020].

Chen, L. et al. FrugalML: How to use ML Prediction APIs more accurately and cheaply. In *NeurIPS*, 2020.

Koenecke, A. et al. Racial Disparities in Automated Speech Recognition. *PNAS*, 117(14):7684–7689, 2020.

Kuznetsova, A. et al. The Open Images Dataset V4. *IJCV*, 128(7):1956–1981, 2020.

Read, J. et al. Multi-label Classification Using Ensembles of Pruned Sets. In *ICDM*, 2008.

Tsoumakas, G. and Vlahavas, I. P. Random *k* -Labelsets: An Ensemble Method for Multilabel Classification. In *ECML*, 2007.

Yao, Y. et al. Complexity vs. performance: empirical analysis of machine learning as a service. In *IMC*, 2017.

Zhang, M. and Zhou, Z. A Review on Multi-Label Learning Algorithms. *IEEE TKDE*, 26(8):1819–1837, 2014.