HOW DID THE MODEL CHANGE? EFFICIENTLY ASSESSING MACHINE LEARNING API SHIFTS

Anonymous authors
Paper under double-blind review

ABSTRACT

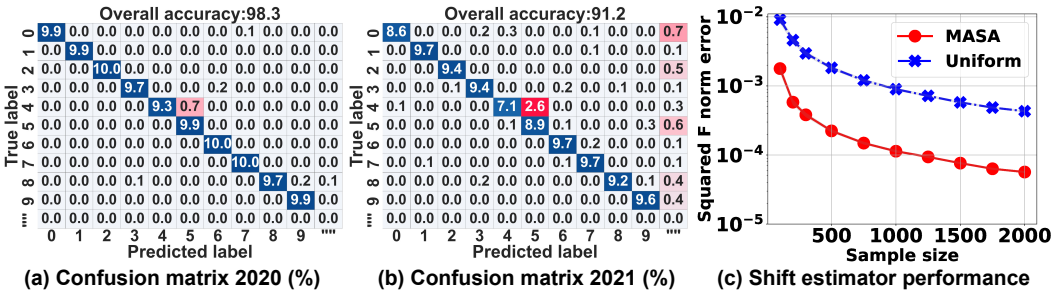
ML prediction APIs from providers like Amazon and Google have made it simple to use ML in applications. A challenge for users is that such APIs continuously change over time as the providers update models, and changes can happen silently without users knowing. It is thus important to monitor when and how much the ML APIs' performance shifts. To provide detailed change assessment, we model ML API shifts as confusion matrix differences, and propose a principled algorithmic framework, MASA, to provably assess these shifts efficiently given a sample budget constraint. MASA employs an upper-confidence bound based approach to adaptively determine on which data point to query the ML API to estimate shifts. Empirically, we observe significant ML API shifts from 2020 to 2021 among 12 out of 36 applications using commercial APIs from Google, Microsoft, Amazon, and other providers. These real-world shifts include both improvements and reductions in accuracy. Extensive experiments show that MASA can estimate such API shifts more accurately than standard approaches given the same budget.

1 INTRODUCTION

Machine learning (ML) prediction APIs have made it dramatically easier to build ML applications. For example, one can use Microsoft text API (Mic, a) to determine the polarity of a text review written by a customer, or Google speech API (Goo, b) to recognize users' spoken commands received by a smart home device. These APIs have been gaining in popularity (MLa; Chen et al., 2020), as they eliminate the need for ML users to train their own models.

Monitoring and assessing the performance of these third-party ML APIs over time, however, is under-explored. ML API providers continuously collect new data or change their model architectures (Qi et al., 2020) to update their services, which could silently help or harm downstream applications' performance. For example, as shown in Figure 1 (a) and (b), we observe a 7% overall accuracy drop of IBM speech API on the AUDIOMNST dataset in March 2021 compared March 2020. In our systematic study of 36 API and dataset combinations, there are 12 cases where the API's performance changed by more than 1% on the same dataset from 2020 to 2021 (sometimes for the worse). Such performance shifts are of serious concern not only because of potential disruptions to downstream tasks, but also because consistency is often required for audits and oversight. Therefore, it is important to precisely assess shifts in an API's predictions over time. Moreover, in this assessment, it is often more informative to quantify how the entire confusion matrix of the API has changed rather than just the overall accuracy. In the IBM case in Figure 1, it is interesting that a major culprit of the drop in performance is the 2021 model mistaking "four" for "five". In other settings, changes in the confusion matrix could still cause issues even if the overall accuracy stays the same.

In this paper, we formalize the problem of assessing API shifts as estimating changes in the confusion matrix on the same dataset in a sample-efficient manner (i.e., with few calls to the API itself to minimize dollar cost). The straightforward approach is to compare the API's prediction on randomly sampled data. However, this can require a large number of API calls to estimate the confusion matrix, which is expensive given that each call costs money. To help address this challenge, we propose MASA, a principled algorithm for ML API shift assessments. MASA efficiently estimates shifts in the API's confusion matrix by clustering the dataset and adaptively sampling data from different clusters to query the API. MASA automates its sampling rate from different data clusters based on the uncertainty in the confusion matrix estimation. For example, MASA may query the ML API on



(a) Confusion matrix 2020 (%) (b) Confusion matrix 2021 (%) (c) Shift estimator performance

Figure 1: ML API shift for IBM speech recognition API on AMNIST, a spoken digit dataset. (a) and (b) give its (normalized) confusion matrix in April 2020 and 2021, respectively. There is an overall 7% accuracy drop. One factor is the 2021 model incorrectly predicting more "four" as "five". (c) Given a sample budget, the proposed MASA can assess the API shift with much smaller error in Frobenius norm compared to standard uniform sampling.

more samples with the true label "four" than "one", if it is less sure about the estimated performance change on the former. Employing an upper-confidence-bound approach to estimate the uncertainties, MASA enjoys a low computation and space cost as well as a fast estimation error rate guarantee.

MASA's adaptive sampling substantially improves the quality of estimation for API shifts. In extensive experiments on real world ML APIs, MASA's assessment error is often an order of magnitude smaller than that of standard uniform sampling with same sample size (e.g., Figure 1 (c)). To reach the same tolerable target estimation error, MASA can reduce the required sample size by more than 50%, sometimes up to 90%.

Contributions. In short, our main contributions include:

1. We demonstrate that commercial Ml APIs can experience significant performance shifts over time, and formulate ML API shift assessments via confusion matrix difference estimation as an important practical problem. We will release the dataset consisting of 1,224,278 samples annotated by ML APIs in different date to stimulate more research on ML API shifts.
2. We propose MASA, an algorithm to assess the ML API performance shifts efficiently. MASA adaptively determines querying the ML API on which data points to minimize the shift estimation error under a sample size constraint. We show that MASA enjoys a low computation cost and performance guarantees.
3. We evaluate MASA on real world APIs from Google, Microsoft, Amazon and other providers for tasks including speech recognition, sentiment analysis, and facial emotion recognition. MASA leads to estimation errors an order of magnitude smaller than standard uniform sampling using the same sample size, or over 90% fewer samples to reach the same tolerable estimation error.

Related Work. **Distribution shifts in ML deployments:** Performance shifts in ML systems have been observed in applications like disease diagnosis (Lipton et al., 2018), facial recognition (Wang et al., 2020), and Inference of molecular structure (Koh et al., 2020). Most of them are attributed to distribution shifts, i.e., the distribution of the test and training datasets are different. Distribution shifts are usually modeled as covariate shifts (Shimodaira, 2000; Sugiyama et al., 2007; Quiñonero-Candela et al., 2009) or label shifts (Lipton et al., 2018; Saerens et al., 2002; Azizzadenesheli et al., 2019; Zhao et al., 2021). API shifts are orthogonal to distribution shifts: instead of attributing the performance shifts to data distribution changes, API shifts concern with ML APIs changes which changes its predictions on the same dataset. The methods for detecting distribution drifts typically rely on changes in data feature statistics and can not detect changes in the API on the same data. To the best of our knowledge, MASA is the first work to systematically investigate ML API shifts.

Deploying and monitoring ML APIs: Several issues in deployed ML APIs have been studied. For example, Buolamwini & Gebru (2018) showed that strong biases toward minority can exist in

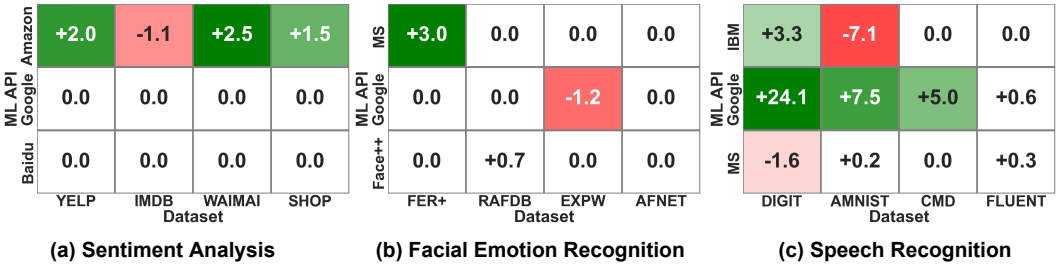


Figure 2: Observed overall accuracy changes. Each row corresponds to an ML API, and each column represents a dataset. The entry is the overall accuracy difference between evaluation in spring 2020 and spring 2021. In 12 out of 36 cases, the API's overall accuracy changed by more than 1%; this includes several cases of substantial drops in performance.

commercial APIs and Ribeiro et al. (2020) revealed that several bugs in commercial APIs can be detected using checklists. Kang et al. (2020) extend program assertions to monitor and improve deployed ML models. Chen et al. (2020) consider the trade-offs between accuracy performance and cost via exploiting multiple APIs. On the other hand, the proposed MASA focuses on estimating (silent) API performance changes cheaply and accurately, which has not been studied before.

Stratified sampling and multi-arm bandits: Stratified sampling has proved to be useful in various domains, such as approximate query processing (Chaudhuri et al., 2007), population mean estimation (Carpentier & Munos, 2011; Carpentier et al., 2015), and complex integration problems (Leprêtre et al., 2017). A common approach is to model stratified sampling as a multi-arm bandit (MAB) problem: view each data partition as an arm, and set the regret as the variance of the obtained estimator. Plenty of algorithms exist for solving standard MAB problems, such as the epsilon greedy algorithm (Slivkins, 2019), upper-confidence-bound approach (Auer et al., 2002), and Thompson sampling method (Russo et al., 2018). While those algorithms aim at playing the best arm as often as possible, stratified sampling's goal is often to estimate the average of all arms, and thus needs to allocate the number of calls for all arms aptly based on their variance. In contrast to estimating the population average in standard stratified sampling, our goal is to simultaneously estimate a matrix whose entries can be correlated. Thus, we have an objective (uncertainty score) that is different from theirs (variance), which also leads to an optimal allocation different from that of standard stratified sampling. This difference requires additional statistical bounds to prove convergence, which we provide in our paper. It also results in a different upper-confidence-bound term and convergence rate compared to standard bandit algorithms.

2 THE API SHIFT PROBLEM

Empirical assessment of ML API shifts. We start by making an interesting observation: *Commercial ML APIs' performance can change substantial over time on the same datasets*. We investigated twelve standard datasets across three different tasks, namely, YELP (Dat, c), IMDB (Maas et al.), WAIMAI (Dat, b), SHOP (Dat, a) for sentiment analysis, FER+ (Goodfellow et al., 2015), RAFDB (Li et al.), EXPW (Zhang et al.), AFNET (Mollahosseini et al., 2019) for facial emotion recognition, and DIGIT (Dat, d), AMNIST (Becker et al., 2018), CMD (Warden, 2018), FLUENT (Lugosch et al.), for speech recognition. For each dataset, we evaluated three commercial ML APIs' accuracy in April 2020 and April 2021. Figure 2 summarizes the overall accuracy changes.

There are several interesting empirical findings. First, API performance changes are quite common. In fact, as shown in Figure 2, API performance changes exceeding 1% occurred in about 33% of all (36) considered ML API-dataset combinations. Since the data distribution remains fixed, such a change is due to ML APIs' updates. Second, the API updates can either help or hurt the accuracy performance depending on the datasets. For example, as shown in Figure 2 (a), the Amazon sentiment analysis API's accuracy increases on YELP, WAIMAI, and SHOP, but decreases on IMDB. In addition, the update of Microsoft facial emotion recognition API only affects performance on the FER+ dataset, as

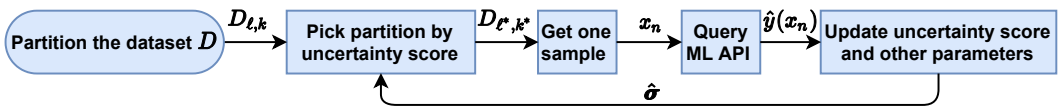


Figure 3: How MASA works. MASA first partitions the dataset. Then it picks which partition to sample based on some uncertainty measurement, queries the ML API on the drawn sample, and uses the API' prediction to update uncertainty and estimated shifts on this partition. This is repeated until the ML API has been queried N times. Finally, the estimated shifts on different partitions are aptly fused to obtain the desired API shifts.

shown in Figure 2 (b). Another interesting finding is that the magnitude of the performance change can be quite different. In fact, most of the accuracy differences are between 1–3%, but on DIGIT dataset, Google's accuracy change is more than 20%.

Fine-grained assessment of API shift as changes in the confusion matrix. Based on feedback from practitioners, accuracy change alone is insufficient, and attribution to per class change is often much more informative (Tsipras et al., 2020; Hou et al., 2019; Luque et al., 2019). Thus, a natural idea is to quantify an ML API's performance by its confusion matrix. We assess the change of the confusion matrix over time as a measure of API shift.

Formally, consider an ML service for a classification task with L labels. For a data point x from some domain $\mathcal{X}$, let $\hat{y}(x) \in [L]$ denote its predicted label on x, and $y(x)$ be the true label. For example, for sentiment analysis, x is a text paragraph, and the task is to predict if the polarity of x is positive or negative. Here $L = 2$, and $\hat{y}(x) = 1$ implies positive predicted label while $y(x) = 2$ indicates negative true label. The confusion matrix is denoted by $\boldsymbol{C} \in \mathbb{R}^{L \times L}$ where $\boldsymbol{C}_{i,j} \triangleq \Pr[y(x) = i, \hat{y}(x) = j]$. Given a confusion matrix of the ML API measured previously (say, a few months ago), $\boldsymbol{C}^o$, the ML API shift is defined as $\Delta\boldsymbol{C} \triangleq \boldsymbol{C} - \boldsymbol{C}^O$.

Using confusion matrix difference to quantify the ML API shift is informative. E.g, the overall accuracy change is simply the trace of $\Delta\boldsymbol{C}$. It also explains which label gets harder or easier for the updated API. Still consider, e.g., sentiment analysis. Given a 2% overall accuracy change, $\Delta\boldsymbol{C}_{1,2} = 1\%$ and $\Delta\boldsymbol{C}_{2,1} = -3\%$ implies that the change is due to predicting less (-3%) negative texts as positive, by sacrificing the accuracy on positive texts slightly (1%). This suggests that the API could have been updated with more negative training texts.

3 MASA: ML API Shift Assessment

Now we present MASA, an algorithmic framework efficiently to assess ML API shifts. Suppose the old confusion matrix $\boldsymbol{C}^o$ and a large labeled dataset D are available. Given a query budget N, our goal is to generate $\Delta\hat{\boldsymbol{C}}$, an estimation of the API shifts as accurately as possible by querying the ML API $\hat{y}(\cdot)$ on N samples drawn from D.

MASA achieves its goal via an adaptive sampling approach (Figure 3). It first divides the given dataset D into several partitions (clusters). Then it adaptively decides which sample to query the ML API in an iterative manner: at each iteration, it selects one data partition based on some uncertainty measure (defined below), and queries the ML API on one sample randomly drawn from this partition. The API's prediction is obtained to update the uncertainty measure as well as the estimated shift $\Delta\hat{\boldsymbol{C}}$. This process is repeated until the ML API has been queried N times or if a stopping rule is reached. We explain each step in detail next.

3.1 Data Partitioning

A key intuition in MASA is that not all samples are equally informative for estimating API shifts. Consider, for example, a vision API makes perfect predictions on "dog" images, and guesses randomly

on "cat" pictures. The "dog" images are less informative, as even a small sample of "dog" queries would tell that there is essentially no confusion for this class. Intuitively, within a sample budget, an estimator with more samples from "cat" pictures should be more accurate overall than that from "dog". Generally, harder images tend to be more informative.

Thus, it is a natural idea to partition all data points based on factors that may correlate with their informativeness, and sample from those partitions separately. In MASA, we use partitions $D_{i,k}$ that each contain the points with true label i and difficulty level k. The difficulty level is an integer indicating how hard it is to predict the data point's label. It needs not be perfect, and can be simply the discretized prediction confidence generated by some simple ML models. A total of L labels and K distinct difficulty labels lead to a total of LK partitions. If the uncertainty or variability of the ML API's prediction on each partition is different, then drawing a different number of samples from each partition may improve the shift assessment performance compared to standard uniform sampling. We verify this empirically in our evaluation (Section 4).

3.2 Budget Allocation Problem

Given the data partition, two questions arise: (i) how many samples should be drawn from each partition, and (ii) how to estimate the ML API shifts given available samples. The second question is relatively straightforward. Note that the API shifts satisfy

$$\begin{aligned}\Delta \boldsymbol{C}_{i,j} &= \Pr[y(x)=i, \hat{y}(x)=j] - \boldsymbol{C}^o_{i,j} = \sum_{i=1}^{L}\sum_{k=1}^{K} \Pr[y(x)=i, \hat{y}(x)=j, x\in D_{i,k}] - \boldsymbol{C}^o_{i,j} \\ &= \sum_{k=1}^{K} \Pr[\hat{y}(x)=j, x\in D_{i,k}] - \boldsymbol{C}^o_{i,j} = \sum_{k=1}^{K} \Pr[x\in D_{i,k}]\Pr[\hat{y}(x)=j | x\in D_{i,k}] - \boldsymbol{C}^o_{i,j}\end{aligned}$$

where the first equation is by definition, the second is due to total probability rule, the third uses the fact that $x \in D_{i,k}$ implies $y(x) = i$, and the last equation applies conditional probability. Here, $\Pr[x \in D_{i,k}]$ is simply ratio of size of partition $D_{i,k}$ and entire dataset D, known a prior. To assess $\Delta\boldsymbol{C}_{i,j}$, we only need to estimate $\Pr[\hat{y}(x) = j|x \in D_{i,k}]$, the predicted label distribution on partition $D_{i,k}$. It can be estimated simply via the frequency of predicting label j among all available samples drawn from $D_{i,k}$.

Now we consider the sample allocation problem. For ease of notation, we denote $\Pr[x_i \in D_{i,j}]$ by $\boldsymbol{p}_{i,k}$, $\Pr[\hat{y}(x) = j|x \in D_{i,k}]$ and its estimation by $\boldsymbol{\mu}_{i,k,j}$ and $\hat{\boldsymbol{\mu}}_{i,k,j}$, respectively. Then for deterministic sample allocations, the squared Frobenius norm error can be written as

$$\begin{aligned}\mathbb{E}\left[\|\Delta\boldsymbol{C} - \Delta\hat{\boldsymbol{C}}\|_F^2\right] &= \sum_{i,j}\mathbb{E}\left(\Delta\boldsymbol{C}_{i,j} - \Delta\hat{\boldsymbol{C}}_{i,j}\right)^2 = \sum_{i,j}\mathbb{E}\left(\sum_k \boldsymbol{p}_{i,k}[\boldsymbol{\mu}_{i,k,j} - \hat{\boldsymbol{\mu}}_{i,k,j}]\right)^2 \\ &= \sum_{i,j,k}\boldsymbol{p}^2_{i,k}\mathbb{E}\left([\boldsymbol{\mu}_{i,k,j} - \hat{\boldsymbol{\mu}}_{i,k,j}]\right)^2\end{aligned}$$

Thus we use the loss $\mathcal{L}(\mathcal{A}, N) \triangleq \sum_{i,j,k}\boldsymbol{p}^2_{i,k}\mathbb{E}\left([\boldsymbol{\mu}_{i,k,j} - \hat{\boldsymbol{\mu}}_{i,k,j}]\right)^2$ to measure the performance of any sample budget allocation algorithm $\mathcal{A}$ using N samples. For any fixed N, our goal is to find a sample budget allocation algorithm A to minimize the loss $\mathcal{L}(A, N)$. Notably, we can generalize it for other scenarios by replacing $(\Delta\boldsymbol{C} - \Delta\hat{\boldsymbol{C}})$ with $\boldsymbol{W} \odot (\Delta\boldsymbol{C} - \Delta\hat{\boldsymbol{C}})$, where $\odot$ is element-wise multiplication and $\boldsymbol{W}$ is an $L \times L$ weight matrix. Different choices of $\boldsymbol{W}$ can penalize the error of each entry in $\Delta\hat{\boldsymbol{C}}$ differently and serve for different purposes. For example, if misclassifying label 1 as label 2 is the only focus, then we can simply set $\boldsymbol{W}_{1,2} = 1$ and $\boldsymbol{W}_{i,j} = 0, \forall(i,j) \neq (1,2)$. If we use the identity matrix as the weight $\boldsymbol{W}$, then it becomes equivalent to estimating the overall accuracy by minimizing the variance. To minimize the loss for the general scenarios, Algorithm 1 (which is explained in the rest of this section) is still applicable by simply multiplying the API selection formula (line 3 in Algorithm 1) with its corresponding weights.

3.3 Uncertainty Score and Optimal Allocation

The optimal sample allocation is directly connected to how informative each data partition is. To see this, let us first introduce the notation of *uncertainty score* for each data partition.

Algorithm 1 MASA's ML API shift assessment algorithm.

Input : ML API $\hat{y}(\cdot)$, query budget N, partitions $D_{i,k}$, $\boldsymbol{p} \in \mathbb{R}^{L \times K}$, $\boldsymbol{C}^o \in \mathbb{R}^{L \times L}$, and $a > 0$

Output : Estimated ML API Shift $\Delta\hat{\boldsymbol{C}} \in \mathbb{R}^{L \times \hat{L}}$

1 Set $\boldsymbol{N} = \boldsymbol{0}_{L \times K}, \hat{\boldsymbol{\mu}} = \boldsymbol{0}_{L \times K \times L}, \hat{\boldsymbol{\sigma}} = \boldsymbol{0}_{L \times K}, \boldsymbol{H} = \boldsymbol{0}_{L \times K \times L}$ ▷ Initialization

2 **for** $n \leftarrow 1$ **to** N **do**

3 $(i^*, k^*) \leftarrow \begin{cases} (i,k), & \textit{if } \boldsymbol{N}_{i,k} < 2 \\ \arg\max_{i,k} \frac{\boldsymbol{p}_{i,k}}{\boldsymbol{N}_{i,k}} \left(\hat{\boldsymbol{\sigma}}_{i,k} + \sqrt[4]{\frac{a}{\boldsymbol{N}_{i,k}}} \right), & \textit{o/w} \end{cases}$ ▷ Determine data partition

4 Sample x_n from D_{i^*,k^*} and query the ML API to obtain $\hat{y}(x_n)$

5 $\boldsymbol{N}_{i^*,k^*} \leftarrow \boldsymbol{N}_{i^*,k^*} + 1$ ▷ Update sample size

6 $\hat{\boldsymbol{\mu}}_{i^*,k^*,j} \leftarrow \hat{\boldsymbol{\mu}}_{i^*,k^*,j} + \frac{\mathbb{1}_{\hat{y}(x_n)=j} - \hat{\boldsymbol{\mu}}_{i^*,k^*,j}}{\boldsymbol{N}_{i^*,k^*}}, \forall j \in [L]$ ▷ Update predicted label distribution

7 $\hat{\boldsymbol{\sigma}}^2_{i^*,k^*} \leftarrow \begin{cases} \frac{1}{2}\boldsymbol{H}_{i^*,k^*,\hat{y}(x_n)}, & \textit{if } \boldsymbol{N}_{i^*,k^*} < 2 \\ \hat{\boldsymbol{\sigma}}^2_{i^*,k^*} + \frac{1 - \frac{\boldsymbol{H}_{i^*,k^*,\hat{y}(x_n)}}{\boldsymbol{N}_{i^*,k^*}-1} - \hat{\boldsymbol{\sigma}}^2_{i^*,k^*}}{\boldsymbol{N}_{i^*,k^*}}, & \textit{o/w} \end{cases}$ ▷ Update uncertainty score

8 $\boldsymbol{H}_{i^*,k^*,\hat{y}(x_n)} \leftarrow \boldsymbol{H}_{i^*,k^*,\hat{y}(x_n)} + 1$ ▷ Update label frequency

9 **end**

10 Return $\Delta\hat{\boldsymbol{C}} \in \mathbb{R}^{L \times L}$ where $\Delta\hat{\boldsymbol{C}}_{i,j} = \sum_{k=1}^{K} \boldsymbol{p}_{i,k}\hat{\boldsymbol{\mu}}_{i,k,j} - \boldsymbol{C}^o_{i,j}, \forall i, j$ ▷ Confusion estimation

Definition 1. $\boldsymbol{\sigma}^2_{i,k} \triangleq (1 - \sum_{j=1}^{L} \Pr^2[\hat{y}(x) = j | x \in D_{i,k}])$ *denotes the uncertainty score of* $D_{i,k}$.

The uncertainty score quantifies how informative each $D_{i,k}$ is by subtracting from 1 the sum of the square of each label's probability mass. The uncertainty score is related to collision entropy (discussed in Appendix A), and determines the optimal allocation as follows.

Lemma 1. *Let* A^* *be the sample allocation algorithm that achieves the smallest expected squared Frobenius norm error. Then the number of samples drawn from* $D_{i,k}$ *by* A^* *is*

$$\boldsymbol{N}^*_{i,k} = \frac{\boldsymbol{p}_{i,k}\boldsymbol{\sigma}_{i,k}}{\sum_{i,k} \boldsymbol{p}_{i,k}\boldsymbol{\sigma}_{i,k}} N$$

Lemma 1 shows that the optimal budget allocation depends on the uncertainty score, but in practice, we do not know the uncertainty score before drawing samples and querying the ML API. Thus, a natural question is how to estimate the uncertainty score $\boldsymbol{\sigma}^2_{i,k}$. Suppose n samples, $x_1, x_2, \cdots, x_n$, are drawn from partition $D_{i,k}$. Then we can estimate $\boldsymbol{\sigma}^2_{i,k}$ by

$$\hat{\boldsymbol{\sigma}}^2_{i,k} \triangleq 1 - \frac{1}{n(n-1)} \sum_{s=1}^{n} \sum_{t:t=1,t\neq s}^{n} \mathbb{1}_{\hat{y}(x_s)=\hat{y}(x_t)} \tag{3.1}$$

3.4 An Uncertainty-aware Adaptive Sampling Algorithm

Now we have a chicken-and-egg problem: estimating the uncertainty scores is needed to find the optimal sample allocation, but sampling from all partitions is needed to estimate their uncertainty scores. To overcome this issue, we adopt an iterative sampling approach, as shown in Algorithm 1. At each iteration, it alternates between (i) uncertainty score-based new sample selection (line 3 - 4) and (ii) uncertainty score and predicted label distribution update using the new sample (line 5 - 8). After querying the ML API N times, the API shifts are obtained by (iii) fusing the estimated predicted label distribution on each partition (line 10). We give the details as follows.

Uncertainty score and predicted label distribution update. After obtaining the predicted label for a sample from partition D_{i^*,k^*}, we need to update (i) the number of samples already drawn from this partition, denoted by $\boldsymbol{N}_{i^*,k^*}$, (ii) the estimated predicted label distribution, denoted by $\hat{\boldsymbol{\mu}}_{i^*,k^*,j}, \forall j$, and (iii) the estimated uncertainty score, $\hat{\boldsymbol{\sigma}}^2_{i^*,k^*}$. For $\boldsymbol{N}_{i^*,k^*}$ and $\hat{\boldsymbol{\mu}}_{i^*,k^*,j}$ (line 5-6), we use standard incremental update approach (Cotton, 1975), which requires constant space and computational cost per iteration. For $\hat{\boldsymbol{\sigma}}^2_{i^*,k^*}$, we additionally maintain the number of label j being predicted among all samples drawn from D_{i^*,k^*}, denoted by $\boldsymbol{H}_{i^*,k^*,j}$ (line 8). This enables a fast incremental update of $\hat{\boldsymbol{\sigma}}^2_{i^*,k^*}$ (line 7).

Uncertainty score-based new sample selection. To determine on which partition to select a new sample, we use an upper-confidence-bound approach on the weighted uncertainty score (second case in line 3), after ensuring two samples have been drawn from each partition (first case in line 3). Two samples are needed for an initial estimation of each partition's uncertainty score. Here, we use a parameter $a > 0$ to balance between exploiting knowledge of uncertainty score ($\hat{\boldsymbol{\sigma}}^2_{i,k}$) and exploring more partitions ($\sqrt[4]{\frac{1}{\boldsymbol{N}_{i,k}}}$).

We quantify the performance of MASA v.s. the optimal allocation algorithm A^* as follows.

Theorem 2. *If $a > 2\log L + \log K + \frac{9}{4}\log N$ and $N > 4LK$, then we have*

$$\mathcal{L}(\text{MASA}, N) - \mathcal{L}(\mathcal{A}^*, N) \leq O(N^{-\frac{5}{4}} \log^{\frac{1}{4}} N)$$

Roughly speaking, Theorem 2 shows that the loss gap between the API shift estimated by MASA and the (unreachable) optimal allocation algorithm ceases in the rate of $N^{-5/4}$. Note that the loss of the optimal allocation decays in the rate of N^{-1}. Thus, as N gets larger and larger, the relative gap becomes more and more negligible.

4 EXPERIMENTS

We apply MASA to estimate the shifts of several real world ML services for various tasks. Our goal is three-fold: (i) understand if and why MASA assess the API shifts efficiently, (ii) examine how much sample cost MASA can reduce compared to standard sampling, and (iii) exploit the trade-offs between estimation accuracy and query cost achieved by MASA. We also study how the hyperparameters affect MASA's performance, left to Appendix C.

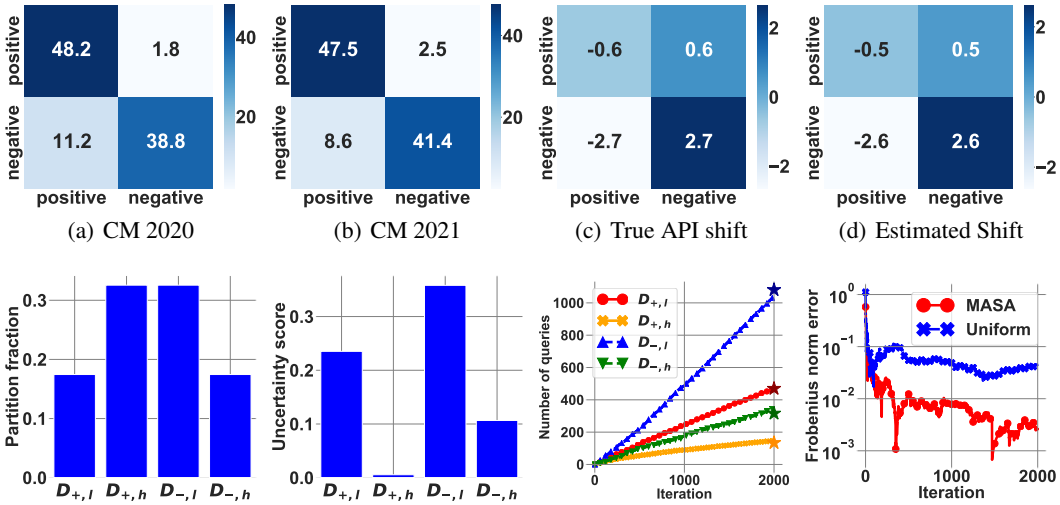
Tasks, ML APIs, and datasets. As shown in Section 2, we have observed 12 of 36 cases where there is a >1% overall accuracy performance change of an ML API. Thus, we focus on evaluating MASA's performance on those 12 cases. Except for case study, all experiments were averaged over 1500 runs. In all tasks, we created partitions using difficulty levels induced by a cheap open source model from GitHub. More details are in Appendix C.

Sentiment analysis: a case study on Amazon API. We start by a case study on Amazon API on a sentiment analysis dataset, YELP to understand MASA's performance. We adopt MASA with sample budget 2000. The dataset is divided into 4 partitions $D_{+,l}, D_{+,h}, D_{-,l}, D_{-,h}$, depending on whether the true label is positive (+) or negative (-), and quality score produced by the 2020 version is lower (l) or higher (h) than the median.

We first note that the API shift gives an interesting explanation to Amazon API's accuracy change. In fact, as shown in Figure 4 (a-c), the accuracy increase is mostly because more texts (2.7%) with negative altitudes are correctly classified. One possible explanation is that the API has been retrained on a dataset with more negative texts. Next, we observe that MASA produces accurate estimation of the API shift by comparing Figure 4 (c) and (d). This is primarily due to (i) that the data partitioning separates more uncertain data from less uncertain ones, and (ii) that adaptive sampling learns the uncertainty level effectively. Figure 4 (e-f) shows that while the partitions' size is similar, their uncertainty scores are different. For example, higher quality score implies a much smaller uncertainty for positive texts ($D_{+,h}$ and $D_{+,l}$ in Figure 4(f)). As shown in Figure 4 (g), MASA indeed learns to utilize such imbalanced uncertainty: its sampling allocation for each partition is quite close to the optimal allocation (dark star point). Finally, it is worth noting that MASA outperforms standard uniform sampling notably, as shown in Figure 4(h). This is because uniform sampling does not exploit the uncertainty of each partition.

Budget savings achieved by MASA. In many applications, it suffices to obtain an estimated API shift close to the true shift, e.g., within a 1% Frobenius norm error. Thus, a natural question arises: *to reach the same tolerable estimation error, how much sampling cost can* MASA *reduce compared to standard sampling approaches?*

To answer this question, we compare MASA with two natural approaches: (i) uniform sampling and (ii) stratified sampling by drawing same number of samples for each true label. For each approach,



(a) CM 2020 (b) CM 2021 (c) True API shift (d) Estimated Shift

(e) Partition Size (f) Uncertainty score (g) Sample allocation (h) Performance

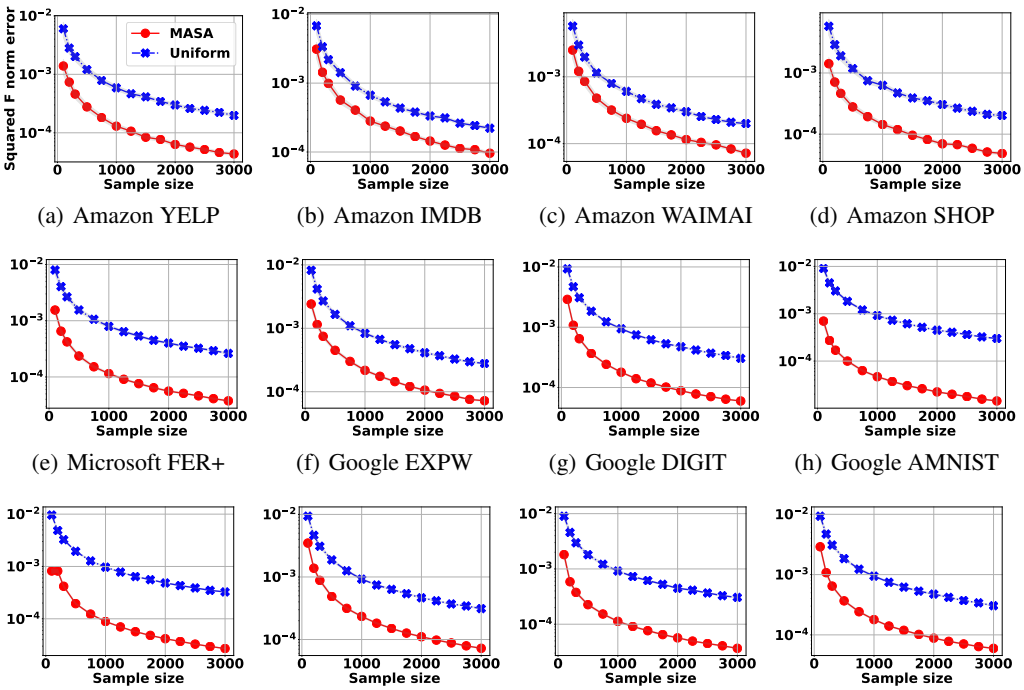
Figure 4: Case study for Amazon API's performance shift on dataset YELP. (a) and (b) give its confusion matrix in spring 2020 and spring 2021, respectively. (c) is their differences, i.e., the API shift. MASA's estimated shift using 2000 samples is in (d). The dataset is divided into 4 partitions based on (i) positive (+) or negative (−) true labels, and (ii) low (l) or high (h) quality score. (e) and (f) give the size and uncertainty score of each partitions. (g) shows MASA's sampling decision per iteration, where the dark dot points represent the (unreachable) optimal sample allocation. (h) reveals its performance.

Table 1: Required sample size to reach 1% Frobnius norm error. Here we compare MASA with uniform (U) sampling and stratified (S) sampling. U and S required very similar sample sizes and are reported in the same column. The sample size is obtained when a 1% Frobenius norm error is achieved with probability 95%.

API;Dataset	Sample size		Save	API;Dataset	Sample size		Save
	MASA	U/S			MASA	U/S	
Amazon;YELP	4.5K	19.7K	77%	IBM;DIGIT	3.6K	17.0K	79%
Amazon;IMDB	10.3K	20.8K	51%	IBM;AMNIST	2.4K	18.5K	87%
Amazon;WAIMAI	7.8K	18.0K	57%	Google;DIGIT	4.2K	17.0K	75%
Amazon;SHOP	4.8K	20.8K	77%	Google;AMNIST	1.1K	18.5K	94%
MS; FER+	2.6K	19.9K	87%	Google;CMD	1.6K	15.2K	89%
Google; EXPW	4.2K	17.9K	77%	MS;DIGIT	3.3K	17.0K	81%

we measure the number of samples needed to reach 1% Frobenius norm error with probability 95%, via an upper bound on the estimated Frobenius error. The details are left to Appendix C. As shown in Table 1, MASA usually requires more than 70% fewer samples to reach such tolerable Frobenius norm error than the uniform and stratified sampling, primarily due to its shift estimation is more accurate. Uniform and stratified sampling required the similar number of samples because the upper bounds on their estimated Frobenius error are similar.

Trade-offs between estimation error and query budget . Next we examine the trade-offs between API shift estimation error and sample size achieved by MASA, shown in Figure 5. We first note that, across all 12 observed API shifts, MASA consistently outperforms standard uniform sampling for any fixed sample size. In fact, the achieved estimation error of MASA is usually an order of magnitude smaller than that of uniform sampling. This verifies that MASA can provide



(a) Amazon YELP (b) Amazon IMDB (c) Amazon WAIMAI (d) Amazon SHOP

(e) Microsoft FER+ (f) Google EXPW (g) Google DIGIT (h) Google AMNIST

(i) Google CMD (j) IBM DIGIT (k) IBM AMNIST (l) Microsoft DIGIT

Figure 5: API shift estimation performance and sample size trade-offs. We compare the expected squared Frobenius norm error of MASA with $K = 3$ partitions versus standard uniform sampling. For any sample size, MASA consistently leads to an estimation error much smaller than uniform sampling across different API and dataset combinations.

more accurate assessments of API shifts in diverse applications. Second, some API shifts are easier to estimate than others. For example, for Google API shift on AMNIST, using 1000 samples already gives an expected squared Frobenius norm error lower than 10^{-4}, while it usually requires 2000 samples for other shifts. This is probably because the skew in its uncertainties among different partitions is more severe than other shifts.

5 Conclusion

In this paper, we identify and formulate the problem of characterizing ML API shifts. Our systematic empirical study has shown that API model updates are frequent, and that some updates can reduce performance substantially. Quantifying such shifts is an important but understudied problem that can greatly affect the reliability of applications using ML-as-a-service. To capture fine-grained change assessment, we model the ML API shifts as confusion matrix differences. Adaptive sampling methods are typically designed for estimating a single scalar (e.g., adaptive population mean) or training a model (e.g., active learning), and thus not directly applicable for API shift estimation. Uniform sampling is natural but requires a large number of samples. Thus, we propose an algorithmic framework, MASA, to adaptively assess the API shifts using as few samples as possible. Our work focuses on estimating changes in the confusion matrix because the confusion matrix is often what is used by practitioners to assess API performance. We acknowledge that confusion matrices are most applicable for classification tasks, and other measures need to be used for more complex APIs (e.g. OCR, NLP). While this is a limitation, classification with a small to moderate number of classes of interest is a common use case for ML APIs, and this is an important starting point since it has not been studied before. We will release our dataset of 1,224,278 samples annotated by commercial APIs in different dates as the first dataset and resource for studying ML API performance shifts.

REFERENCES

Amazon Comprehend API. https://aws.amazon.com/comprehend. [Accessed March-2020 and March-2021].

Baidu API. https://ai.baidu.com/. [Accessed March-2020 and March-2021].

SHOP dataset, https://github.com/SophonPlus/ChineseNlpCorpus/tree/master/datasets/online_shopping_10_cats. a. URL https://github.com/SophonPlus/ChineseNlpCorpus/tree/master/datasets/online_shopping_10_cats.

WAIMAI dataset, https://github.com/SophonPlus/ChineseNlpCorpus/tree/master/datasets/waimai_10k. b. URL https://github.com/SophonPlus/ChineseNlpCorpus/tree/master/datasets/waimai_10k.

YELP dataset, https://www.kaggle.com/yelp-dataset/yelp-dataset. c. URL https://www.kaggle.com/yelp-dataset/yelp-dataset.

DIGIT dataset, https://github.com/Jakobovski/free-spoken-digit-dataset. d. URL https://github.com/Jakobovski/free-spoken-digit-dataset.

Face++ Emotion API. https://www.faceplusplus.com/emotion-recognition/. [Accessed March-2020 and March-2021].

Google NLP API. https://cloud.google.com/natural-language. [Accessed March-2020 and March-2021].

Google Vision API. https://cloud.google.com/vision, a. [Accessed March-2020 and March-2021].

Google Speech API. https://cloud.google.com/speech-to-text, b. [Accessed March-2020 and March-2021].

IBM Speech API. https://cloud.ibm.com/apidocs/speech-to-text. [Accessed March-2020 and March-2021].

Machine Learning as a Service Market Report . https://www.mordorintelligence.com/industry-reports/global-machine-learning-as-a-service-mlaas-market.

Microsoft computer vision API. https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision, a. [Accessed March-2020 and March-2021].

Microsoft speech API. https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text, b. [Accessed March-2020 and March-2021].

K.B. Athreya and S.N. Lahiri. *Measure Theory and Probability Theory*. Springer, 2006.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256, 2002. doi: 10.1023/A:1013689704352. URL https://doi.org/10.1023/A:1013689704352.

Kamyar Azizzadenesheli, Anqi Liu, Fanny Yang, and Animashree Anandkumar. Regularized learning for domain adaptation under label shifts. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJl0r3R9KX.

Sören Becker, Marcel Ackermann, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Interpreting and explaining deep neural networks for classification of audio signals. *CoRR*, abs/1807.03418, 2018.

Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In Sorelle A. Friedler and Christo Wilson (eds.), *Conference on Fairness, Accountability and Transparency, FAT 2018, 23-24 February 2018, New York, NY, USA*, volume 81 of *Proceedings of Machine Learning Research*, pp. 77–91. PMLR, 2018. URL http://proceedings.mlr.press/v81/buolamwini18a.html.

Alexandra Carpentier and Rémi Munos. Finite time analysis of stratified sampling for monte carlo. In John Shawe-Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pp. 1278–1286, 2011. URL https://proceedings.neurips.cc/paper/2011/hash/872488f88d1b2db54d55bc8bba2fad1b-Abstract.html.

Alexandra Carpentier, Rémi Munos, and András Antos. Adaptive strategy for stratified monte carlo sampling. *J. Mach. Learn. Res.*, 16:2231–2271, 2015. URL http://dl.acm.org/citation.cfm?id=2886821.

Surajit Chaudhuri, Gautam Das, and Vivek R. Narasayya. Optimized stratified sampling for approximate query processing. *ACM Trans. Database Syst.*, 32(2):9, 2007. doi: 10.1145/1242524.1242526. URL https://doi.org/10.1145/1242524.1242526.

Lingjiao Chen, Matei Zaharia, and James Y. Zou. Frugalml: How to use ML prediction apis more accurately and cheaply. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Ira W. Cotton. Remark on stably updating mean and standard deviation of data. *Commun. ACM*, 18 (8):458, 1975. doi: 10.1145/360933.360981. URL https://doi.org/10.1145/360933.360981.

Ian J. Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron C. Courville, Mehdi Mirza, Benjamin Hamner, William Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Tudor Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio. Challenges in representation learning: A report on three machine learning contests. *Neural Networks*, 64:59–63, 2015. doi: 10.1016/j.neunet.2014.09.005. URL https://doi.org/10.1016/j.neunet.2014.09.005.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 831–839. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00092. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Hou_Learning_a_Unified_Classifier_Incrementally_via_Rebalancing_CVPR_2019_paper.html.

Daniel Kang, Deepti Raghavan, Peter Bailis, and Matei Zaharia. Model assertions for monitoring and improving ML models. In Inderjit S. Dhillon, Dimitris S. Papailiopoulos, and Vivienne Sze (eds.), *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org, 2020. URL https://proceedings.mlsys.org/book/319.pdf.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. *CoRR*, abs/2012.07421, 2020. URL https://arxiv.org/abs/2012.07421.

Florian Leprêtre, Fabien Teytaud, and Julien Dehos. Multi-armed bandit for stratified sampling: Application to numerical integration. In *Conference on Technologies and Applications of Artificial Intelligence, TAAI 2017, Taipei, Taiwan, December 1-3, 2017*, pp. 190–195. IEEE Computer Society, 2017. doi: 10.1109/TAAI.2017.34. URL https://doi.org/10.1109/TAAI.2017.34.

Shan Li, Weihong Deng, and JunPing Du. Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. In *CVPR 2017*.

Zachary C. Lipton, Yu-Xiang Wang, and Alexander J. Smola. Detecting and correcting for label shift with black box predictors. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3128–3136. PMLR, 2018. URL http://proceedings.mlr.press/v80/lipton18a.html.

Loren Lugosch, Mirco Ravanelli, Patrick Ignoto, Vikrant Singh Tomar, and Yoshua Bengio. Speech model pre-training for end-to-end spoken language understanding. In *Interspeech 2019*.

Amalia Luque, Alejandro Carrasco, Alejandro Martín, and Ana de las Heras. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognit.*, 91:216–231, 2019. doi: 10.1016/j.patcog.2019.02.023. URL https://doi.org/10.1016/j.patcog.2019.02.023.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Human Language Technologies, ACL 2011*.

Ali Mollahosseini, Behzad Hasani, and Mohammad H. Mahoor. Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Trans. Affect. Comput.*, 10(1):18–31, 2019. doi: 10.1109/TAFFC.2017.2740923. URL https://doi.org/10.1109/TAFFC.2017.2740923.

Haode Qi, Lin Pan, Atin Sood, Abhishek Shah, Ladislav Kunc, and Saloni Potdar. Benchmarking intent detection for task-oriented dialog systems. *CoRR*, abs/2012.03929, 2020. URL https://arxiv.org/abs/2012.03929.

Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. *Covariate Shift by Kernel Mean Matching*, pp. 131–160. 2009.

Marco Túlio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with checklist. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 4902–4912. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.442. URL https://doi.org/10.18653/v1/2020.acl-main.442.

Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A tutorial on thompson sampling. *Found. Trends Mach. Learn.*, 11(1):1–96, 2018. doi: 10.1561/2200000070. URL https://doi.org/10.1561/2200000070.

Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Comput.*, 14(1):21–41, 2002. doi: 10.1162/089976602753284446. URL https://doi.org/10.1162/089976602753284446.

Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000. ISSN 0378-3758. doi: https://doi.org/10.1016/S0378-3758(00)00115-4. URL https://www.sciencedirect.com/science/article/pii/S0378375800001154.

Aleksandrs Slivkins. Introduction to multi-armed bandits. *Found. Trends Mach. Learn.*, 12(1-2):1–286, 2019. doi: 10.1561/2200000068. URL https://doi.org/10.1561/2200000068.

Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis (eds.), *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pp. 1433–1440. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper/2007/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract.html.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. From imagenet to image classification: Contextualizing progress on benchmarks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9625–9635. PMLR, 2020. URL http://proceedings.mlr.press/v119/tsipras20a.html.

Zhongyuan Wang, Guangcheng Wang, Baojin Huang, Zhangyang Xiong, Qi Hong, Hao Wu, Peng Yi, Kui Jiang, Nanxi Wang, Yingjiao Pei, Heling Chen, Yu Miao, Zhibing Huang, and Jinbi Liang. Masked face recognition dataset and application. *CoRR*, abs/2003.09093, 2020. URL https://arxiv.org/abs/2003.09093.

Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *CoRR*, abs/1804.03209, 2018. URL http://arxiv.org/abs/1804.03209.

Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Learning social relation traits from face images. In *ICCV 2015*.

Eric Zhao, Anqi Liu, Animashree Anandkumar, and Yisong Yue. Active learning under label shift. In Arindam Banerjee and Kenji Fukumizu (eds.), *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pp. 3412–3420. PMLR, 2021. URL http://proceedings.mlr.press/v130/zhao21b.html.