Model ChangeLists: Characterizing Updates to ML Models

Sabri Eyuboglu
eyuboglu@stanford.edu
Stanford University
Stanford, California, United States

Karan Goel
Stanford University
Stanford, California, United States

Arjun Desai
Stanford University
Stanford, California, United States

Lingjiao Chen
Stanford University
Stanford, California, United States

Mathew Monfort
Amazon Web Services
Seattle, Washington, United States

Christopher Ré
Stanford University
Stanford, California, United States

James Zou
Stanford University
Stanford, California, United States

ABSTRACT

Updates to Machine Learning as a Service (MLaaS) APIs may affect downstream systems that depend on their predictions. However, performance changes introduced by these updates are poorly documented by providers and seldom studied in the literature. As a result, API producers and consumers are left wondering: *do model updates introduce performance changes that could adversely affect users' system?* Ideally, producers and consumers would have access to a detailed `ChangeList` specifying the slices of data where model performance has improved and degraded since the update. But, producing a `ChangeList` is challenging because it requires (1) discovering slices in the absence of detailed annotations or metadata, (2) accurately attributing coherent concepts to the discovered slices, and (3) communicating them to the user in a digestable manner. In this work, we demonstrate, discuss, and critique one approach for building, verifying, and releasing `ChangeLists` that aims to address these challenges. Using this approach, we analyze six real-world MLaaS API updates including GPT-3 and Google Cloud Vision. We produce a prototype `ChangeList` for each, identifying over 100 coherent data slices on which the model's performance changed significantly. Notably, we find 63 instances where an update improves performance globally, but hurts performance on a coherent slice – a phenomenon not previously documented at scale in the literature. Finally, with diverse participants from industry, we conduct a think-aloud user study that explores the importance of releasing `ChangeLists` and highlights the strengths and weaknesses of our approach. This serves to validate some parts of our approach and uncover important areas for future work.

ACM Reference Format:
Sabri Eyuboglu, Karan Goel, Arjun Desai, Lingjiao Chen, Mathew Monfort, Christopher Ré, and James Zou. 2024. Model ChangeLists: Characterizing Updates to ML Models. In *The 2024 ACM Conference on Fairness, Accountability, and Transparency (FAccT '24), June 03–06, 2024, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 22 pages. https://doi.org/10.1145/3630106.3659047

1 INTRODUCTION

Modern software systems often depend on Machine Learning as a Service (MLaaS) APIs developed by cloud providers (*e.g.* AWS, GCP, Azure) or research organizations (*e.g.* OpenAI, HuggingFace). The models behind these APIs are periodically updated and new versions are released. However, to a consumer of an API, how a new update will affect the workings of their broader system is typically unclear. Consider, for example, a newspaper that uses an image tagging API to source archival photos for retrospective stories [30]. Updates to the underlying model could lead to unexpected changes in the workflow of photo editors and journalists who rely on the system.

MLaaS producers (*e.g.* Google, OpenAI) rarely provide transparent evaluations of their updates, and those that do focus on global metrics and vague notions of improvement. Release notes from API producers are terse and provide little information. For example, Microsoft's Vision API (Feb '22 update) only notes "general performance and AI quality improvements" [52], and OpenAI's post on the difference between two versions of GPT-3 cites only two concrete examples of model outputs [56].

These release notes tell an incomplete story: saying that one model improves on another obscures the fact that models may perform very differently on fine-grained slices of data [15, 68]. Returning to the newspaper example described above, image tagging performance after a model update may improve globally while deteriorating on historic photos – the kind of photos commonly found in the newspaper's archives. Without more detailed evaluations, both API producers and consumers are left wondering:

Do model updates introduce changes that adversely affect users' systems?

While many studies include detailed comparisons of MLaaS APIs [3, 26, 27, 64, 68], they lack comparisons of the *same* API before and after an update. Recent work shows that API updates can lead to performance drops on benchmarks [4], but the analysis is limited to simple tasks and global measurements.

API Update	Global Metrics		Slice Counts	Most/Least Improved	
Microsoft Computer Vision	**Accuracy**	μ: ↑ 0.012	↑ 81 slices (51)	μ: ↑ 0.55 (0.51, 0.59)	*"Street name signs"*
`Nov 2020 → Feb 2022`		σ: 0.166	↓ 28 slices (16)	μ: ↓ 0.391 (-0.45, -0.32)	*"Bathrooms with visible toilet"*
Google Cloud Vision	**Accuracy**	μ: ↑ 0.007	↑ 66 slices (42)	μ: ↑ 0.864 (0.83, 0.9)	*"Motorcycle wheel"*
`Nov 2020 → Feb 2022`		σ: 0.172	↓ 47 slices (31)	μ: ↓ 0.674 (-0.73, -0.63)	*"Beds in hotel rooms"*
EveryPixel	**Accuracy**	μ: ↓ 0.005	↑ 70 slices (47)	μ: ↑ 0.922 (0.89, 0.94)	*"Cats near windows and doors"*
`Nov 2020 → Feb 2022`		σ: 0.166	↓ 31 slices (16)	μ: ↓ 0.158 (-0.19, -0.12)	*"Sinks near cats"*
OpenAI Text Davinci (GPT-3)	**F1-Score**	μ: ↑ 0.038	↑ 26 slices (14)	μ: ↑ 0.151 (0.06, 0.23)	*"Geography"*
`Mar 2022 → Nov 2022`		σ: 0.332	↓ 5 slices (0)	μ: ↓ 0.048 (-0.1, 0.02)	*"Basketball"*
Cohere XLarge	**F1-Score**	μ: ↑ 0.023	↑ 28 slices (10)	μ: ↑ 0.124 (0.01, 0.25)	*"Baseball"*
`Jun 2022 → Nov 2022`		σ: 0.428	↓ 3 slices (0)	μ: ↓ 0.041 (-0.14, 0.04)	*"Geography"*
AI21 Jurassic-1 Grande	**F1-Score**	μ: ↑ 0.059	↑ 28 slices (21)	μ: ↑ 0.177 (0.14, 0.22)	*"Who question"*
`May 2022 → November 2022`		σ: 0.365	↓ 3 slices (0)	μ: ↓ 0.016 (-0.08, 0.07)	*"Medicine"*

Figure 1: Overview of Updates. We produce `ChangeLists` for three image tagging API updates (top) [18, 29, 51] and three question answer API updates (bottom) [11, 45, 56]. From left to right, we provide the dates of the update, the global performance shift across the dataset (μ), the global performance inconsistency (σ), the count of slices where performance improved and degraded (statistically significant improvements and degradations in parentheses), and the slices with the largest improvements and degradations (95% bootstrap confidence intervals in parentheses).

Answering this question would be easier if producers released more detailed reports. To formalize this, we introduce the notion of a `ChangeList`: an interactive report specifying the slices of data where model performance has changed. `ChangeLists` allow users to explore how the model's behavior has changed on the slices most important to their system. For the example above, the newspaper should be able to draw conclusions like: "the updated API detects objects in historic photos with 10% lower recall." Such conclusions would inform decisions around whether or not to integrate the update. However, producing a comprehensive `ChangeList` is difficult due to 3 main challenges:

(1) For complex data types like images and natural language, the set of slices that partition the data is extremely large and unknown a priori. *How can we gather coherent slices that explain the change?*
(2) When interpreting slices, we typically attribute concepts (*e.g. historic*) to them. However, if the slice was discovered automatically, it may not align perfectly with a concept, leading to false conclusions about performance on the concept. *How do we quickly perform accurate attribution?*
(3) The number of slices with significant changes can be very large, and not all changes will be relevant to all users. *How do we help users surface slices most important to their system?*

To better understand these challenges, we demonstrate and discuss one possible approach for building, verifying and releasing `ChangeLists` for model updates. Our approach consists of three phases:

(1) **Discovery**: First, we gather candidate slices by automatically *discovering* candidate slices for the `ChangeList`. We use contrastively-learned embeddings and a simple mixture model to identify slices of data where the models differ [65]. Optionally, discovered slices can be refined using natural language search powered by contrastively-learned embeddings and open-source language models [9]. This is a straightforward adaptation of a recently proposed slice discovery method [19]. There are many other methods that could also be used to automatically discover slices [16, 23, 34, 37, 54, 74].
(2) **Attribution**: Next, we ascribe concepts to the discovered slices. Via an interactive process termed *micro-labeling*, we verify the accuracy of the attributions and dynamically correct them. Contrastively-learned embeddings (*e.g.* CLIP) are used to guide an importance sampler [58] that surfaces a small number of examples for labeling. Labeled examples are then used to estimate the precision and recall of the user attributions, and to update slices to reflect label feedback.
(3) **Release**: Finally, to help users understand model updates, we prototype a simple web interface. The slices in the `ChangeList` are indexed by cross-modal embeddings, and are therefore easily searchable by text or image. Further, if the `ChangeList` is missing slices, they can initiate discovery and attribution to edit the `ChangeList`.

We apply this approach to study updates to six popular machine learning APIs: three large language models (*e.g.* GPT-3) and three image tagging models (*e.g.* Google Cloud Vision).

We produce one `ChangeList` per API update, which together include over 100 coherent slices on which the model's performance changed significantly over time. These slices were not annotated in the dataset and were discovered using phases 1 and 2 of the approach described in Section 3. Of these, there are 63 slices in the `ChangeLists` on which an API update introduced a statistically significant degradation in performance. For example, between 2020 and 2022, the accuracy of a model from Google Cloud Vision on the task of tagging "people" degraded by 17.7%-points for black and white images. This phenomenon, where an update improves performance globally but hurts performance on a coherent slice, has not been documented at this scale in the literature and underscores

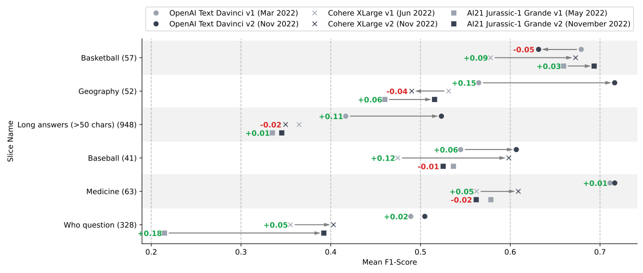


Figure 2: Shifts in Question Answering Performance on Slices. For six slices, we show the change in F1-score for all three question answering APIs. The slices with the largest improvement and degradation for each API are shown. The x-axis shows the F1-score. The y-axis shows the name ascribed to the slice in attribution and its size in parentheses.

the importance of releasing detailed `ChangeLists` alongside model updates.

Finally, to better understand the role `ChangeLists` could play in practice, we carried out a think-aloud user study with a diverse pool of participants including engineers, product managers, researchers, and open source developers working on or with MLaaS APIs. In the interviews, participants emphasized the need for transparent evaluations of API updates. They also commented on strengths of our approach including the discovery and attribution phases, while also highlighting important areas for improvement. Participants suggested that `ChangeLists`, were they deployed, would impact day-to-day processes around how people release and use MLaaS APIs. Altogether, our findings underscore the importance of releasing slice-based `ChangeLists` and provides insights for the development `ChangeLists` into the future.

1.1 Prior Work

Evaluating MLaaS APIs. A growing number of publications include evaluations of MLaaS APIs. Some evaluate a single API in depth [32]. Others compare several different APIs on the same task [33, 67, 77]. For example, Chen *et al.* compare APIs from different producers and demonstrate that performance varies by class [6]. In the language modeling community, there have been several large-scale efforts to evaluate MLaaS APIs [44, 73]. Several studies discuss significant racial disparities in the performance of MLaaS APIs [3, 39, 50]. More generally, evaluation frameworks like Checklist and RobustnessGym applied to MLaaS APIs [27, 68] demonstrate an array of vulnerabilities not discernible with standard evaluations. While some of these studies compare APIs from different producers, few compare different versions of an API from the *same* API. Recently, Chen et al. [4] showed that the accuracy of ML APIs sometimes changes after an update. This analysis, which is most similar to our own, is limited to simple classification tasks and does not consider error consistency or slice-level differences in performance.

Comparing Machine Learning Models. Prior studies have compared machine learning models by measuring the consistency of the errors made by different image classifiers [21, 23, 24, 28, 47, 76]. For example, Mania et al. [47] measure the consistency of errors made by different ImageNet classifiers with the same accuracy, showing that error consistency is significantly higher than would be expected if predictions from different models were independent. Building on this, recent work explores how differences in model initialization and architecture affect the consistency of errors [21, 28]. Instead of using a fixed set of test inputs, others generate new inputs where models disagree [43, 60, 75] or compare outputs of explanation methods [35]. Finally, recent work proposes comparing the output of post-hoc explanation methods applied to different models. For example, Geirhos *et al.* measure the consistency of predictions using Cohen's kappa coefficient. In a study of object recognition models, they find that the error consistency between convolutional neural networks (CNN) is higher than the consistency between CNNs and humans[23]. Similarly, Mania *et al.* measure the consistency of errors made by different ImageNet classifiers with the same accuracy, showing that error consistency is significantly higher than would be expected if predictions from different models were independent [47]. Fort *et al.* show that two models with the same architecture but different random initializations exhibit significant inconsistency in their predictions [21]. Building on this, Gontijo-Lopes *et al.* show that the inconsistency in the errors of two models grows as their training methodologies diverge [28]. While these studies measure the inconsistency of model predictions, they don't explore what this inconsistency means for performance differences on fine-grained slices.

Model and Dataset Reporting. Our work builds on a recent efforts to standardize and improve the documentation of models and datasets in the machine learning community [12, 22, 25, 53, 63, 69, 70]. Most similar to ours is the work of Crisnan *et al.*, who develop interactive design inquiry into interactive model cards [12].

2 MEASURING GLOBAL CHANGES IN REAL API UPDATES

We first introduce *change metrics*, summary statistics that describe the effect of a model update on performance. Our metrics measure (1) the *performance shift* due to the update and (2) the *inconsistency* of this shift across the data. We then apply these metrics to six real API updates, motivating the need for fine-grained `ChangeLists` Section 3.

Preliminaries. Consider a supervised learning setup where each *example* (X, Y) is composed of an *input* $X \in \mathcal{X}$ (*e.g.* an image) and a *target* $Y \in \mathcal{Y}$ (*e.g.* a binary label). Assume we have a loss function (or point-wise metric) $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$. Additionally, we have black-box access to two models trained for this task $v^{[1]}, v^{[2]} : \mathcal{X} \rightarrow \mathcal{Y}$ – e.g. these models could serve predictions for the *same* MLaaS API at different points in time: $v^{[1]}$ before an update and $v^{[2]}$ after, or they can correspond to two different models addressing the same task. To compare the models, we collect their predictions $\hat{y}_i^{[j]} := v^{[j]}(x_i)$ on a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n} \sim \mathcal{P}(X, Y)$, where n is the total number of examples in the dataset.

Change Metrics. We define two change metrics in terms of $D = \ell(v^{[1]}(X), Y) - \ell(v^{[2]}(X), Y)$, the difference in loss between the models,

(1) **Performance Shift** (*on average, did the model improve as a result of the update?*). This metric estimates the expected difference in losses,

$$\mu_\ell = \mathbb{E}[\ell(v^{[1]}(X), Y) - \ell(v^{[2]}(X), Y)] = \mathbb{E}[D]. \quad (1)$$

Positive values of μ indicate that the model improved after the update.

(2) **Performance Inconsistency** (*is the shift in performance inconsistent across the dataset?*). This metric estimates the standard deviation of the difference between the losses,

$$\sigma_\ell^2 = \mathrm{Var}[\ell(v^{[1]}(X), Y) - \ell(v^{[2]}(X), Y)] = \mathrm{Var}[D]. \quad (2)$$

Larger values of σ_ℓ indicate that the models frequently disagree, and the shift is not consistent. This metric is inspired by prior consistency metrics [23, 28].

Example: zero-one loss. For the special case of the zero-one loss, $\ell_{01} : \mathcal{Y} \times \mathcal{Y} \rightarrow \{0, 1\}$, we have

$$\mu_{01} = \frac{1}{n}\sum_{i=1}^{n}([\hat{y}_i^{[1]} = y_i] - [\hat{y}_i^{[2]} = y_i]), \quad \sigma_{01}^2 = \frac{1}{n}\sum_{i=1}^{n}[\hat{y}_i^{[2]} \neq \hat{y}_i^{[1]}] - \mu_{01}^2.$$

Observe that μ_{01} is simply the difference in accuracy between the models, while σ_{01}^2 measures the disagreement between the models that is left over after accounting for some of the performance shift. The maximum $\sigma_{01} = 1$ occurs when both models have the same accuracy but disagree everywhere.

Discussion. These metrics allow us to measure when users should be cautious in using an updated model. With positive performance shift and no inconsistency, model updates can be integrated by users safely. However, high performance inconsistency even absent performance shift is concerning, since the update may disproportionately hurt performance on data important to the user's application.

2.1 Global Analysis of API Updates

Next, we use the change metrics we introduced to analyze six real API updates: three updates on an image tagging task, and three on a question answering task. We briefly discuss these tasks, with full details in Appendix C.

Image Tagging. In image tagging, the input X is an image and category (*e.g. horse*) pair and the target $Y \in \{0, 1\}$ is a binary label indicating whether an object of the category is in the image. We consider updates to three image tagging APIs: *Microsoft* Computer Vision API, *Google [29]* Cloud Vision API, and *EveryPixel [18]* Image Keywording Service. The predictions are sourced from History of APIs [5], a longitudinal database of API predictions that includes predictions for the LVIS dataset in November 2020 and February 2022. We analyze global performance changes using the point-wise zero-one loss $\ell_{01}(y, \hat{y}^{[i]}) = \mathbf{1}[y = \hat{y}^{[i]}]$, as well as metrics that are not point-wise: recall, precision, and F1-score.

We measure performance on the Large Vocabulary Instance Segmentation (LVIS) dataset, a relabeling of the COCO dataset [31, 46] that reflects the breadth of categories output by image tagging APIs (n = 1,577,603 across 1,203 categories). We additionally process the API outputs to map to labels in LVIS (details in Appendix C).

Results. From Nov '20 to Feb '22, *Google* and *Microsoft* saw shifts in mean zero-one loss μ_{01} of +0.007 and +0.012 respectively (+0.04 and +0.059 F1), while *EveryPixel* saw a small degradation in μ_{01} of −0.005 (−0.029 F1). However, these shifts tell only a partial story: all three exhibit non-zero performance inconsistency ($\sigma_{01} > 0.15$). To put this into context, the predictions of the Google API ($\sigma_{01} = 0.172$ and $\sigma_{01}|Y = 0.326$) changed on 10 + % of positive examples.

Question Answering. In question answering, the input X is a natural language question optionally accompanied by context, and the target Y is a list of gold answers. We consider updates to three large language model APIs: *OpenAI's Text DaVinci* (*a.k.a* GPT-3) [1], *Cohere's X-Large generative model* [11], and *AI21's Jurassic-1 Grande* [45]. The predictions are sourced from HELM, a database of API predictions [44]. Following prior work in QA [42, 44, 66], we measure global performance changes using point-wise F1-score ℓ_{F1}. This metric awards partial credit for string overlap by computing the maximum F1-score between the tokens in the predicted text and any one of the gold answers.

We measure performance on an ensemble of question answering datasets: NaturalQA [42], NarrativeQA [38], BoolQ [10], and QUAC [7] (n = 4,470). This ensemble includes both *open-book* and *closed-book* QA.

Results. All three updates led to increases in mean F1-score, with μ_{F1} of +0.038, +0.023, and +0.059 for OpenAI, Cohere, and AI21 respectively. Again, these shifts tell a partial story: all updates exhibit high performance inconsistency ($\sigma_{F1} > 0.3$). Indeed, OpenAI's F1-score improved on 21.2% of examples after the update, but degraded on 14%. This highlights that the global change metric, μ_{F1} = +0.038, fails to explain the API's behavior changes between updates.

Summary. Overall, both image tagging and question answering APIs showed improvements in mean performance metrics, but these improvements were not uniform across all examples. Instead, these APIs exhibited non-zero performance inconsistency, indicating changes in performance on a significant portion of inputs between updates. This highlights the limitations of considering only global changes in performance when evaluating API updates, and motivates the introduction of `ChangeLists` next.

3 CHARACTERIZING MODEL UPDATES IN DETAIL

These findings highlight the importance of producing a more fine-grained understanding of updates. This motivates our key proposal: the introduction of a `ChangeList` (Section 3.1) to explain the observed performance shift and inconsistency using changes in fine-grained slices, and an interactive process to build them (Section 3.2).

3.1 The `ChangeList`: A Fine-Grained Characterization of Model Updates

For a user, the decision on whether to use an updated model requires understanding the data examples that account for performance shift and inconsistency. Users seek explanations that focus on the data important to their application. We formalize this relationship through *slices* of data important to users, and define `ChangeLists` in terms of these slices.

Slices (S). A *slice* is a subset of data examples that share something in common e.g. in object recognition, the set of images with dim lighting would constitute a slice. Formally, we represent a slice with a random variable $S \in \{0, 1\}$ and a set of k slices with $\mathrm{S} = \{S^{(j)}\}_{j=1}^{k} \in \{0, 1\}^k$, with joint distribution $P(X, Y, \mathrm{S})$ over inputs, targets and slices. Each example has a realization of the slice random variables $\{s_i^{(j)}\}_{i=1}^{n}$. If $s_i^{(j)} = 1$, then example (x_i, y_i) is in slice $S^{(j)}$. In practice, datasets do not include realizations for all possible slices, e.g. not including annotations for dim lighting. Intuitively, we would like a ChangeList to present users with slices alongside human-readable descriptions and metrics quantifying how their performance has changed. We define these next.

Slice Attributions (A). Define random variable $A^{(j)} \in \{0, 1\}$ to represent the presence of a text attribution for slice $S^{(j)}$ (e.g. *"dim lighting"*), with attributions $\mathbf{A} = \{A^{(j)}\}_{j=1}^{k}$ corresponding to S and example level attribute realizations $\{a_i^{(j)}\}_{i=1}^{n}$. If $a_i^{(j)} = 1$, then example (x_i, y_i) satisfies attribution $A^{(j)}$. Typically, these realizations are unknown, and only the text attribution $A^{(j)}$ will be given.

Slice Change Metrics (M). Given a slice $S^{(j)}$, denote change metrics $\mu_\ell^{(j)}, \sigma_\ell^{(j)}$ for loss functions $\ell_1, \ldots, \ell_r$, with the set of change metrics for S denoted by $\mathbf{M} = \{\mu_{\ell_1}^{(j)}, \sigma_{\ell_1}^{(j)}, \ldots, \mu_{\ell_r}^{(j)}, \sigma_{\ell_r}^{(j)}\}_{j=1}^{k}$.

We are now ready to define a ChangeList using these concepts.

Definition (ChangeList $\mathfrak{C}$). *Given dataset $\mathcal{D}$ and models $v^{[1]}, v^{[2]}$, a* ChangeList *is a collection of slices* S *along with their corresponding descriptions* A *and change metrics* M.

What constitutes a good ChangeList? We discuss six criteria that we expect will be desired by users of ChangeLists. These desiderata are not exhaustive, and we expect more to emerge as ChangeLists are adopted into wider practice.

(1) **Diversity (of S).** Different users have different slices of interest e.g. decorators may tag images of homes, while doctors may tag hospital images. ChangeLists should contain a diversity of slices to reflect this.
(2) **Coverage (of σ_ℓ with S).** The slices S should together explain the inconsistency σ_ℓ. The explanatory power of S can be measured by the coefficient of determination $r^2 = 1 - \frac{1}{\sigma_\ell^2}\mathbb{E}[(D - f(\mathbf{S}))^2]$, where $f(\mathbf{s}) = a + \mathbf{b}^T\mathbf{s}$ is a function fit by performing a linear regression of D on S.
(3) **Alignment (of A with S).** Attributions should align with the examples in each slice, and users should be able to read these attributions to understand the content of each slice.
(4) **Relevance (of M).** Change metrics reported in the ChangeList should be chosen to be relevant to the tasks for which the models are to be used.
(5) **Navigability (of S).** Users of ChangeLists should be able to search over information in the ChangeList, including global change metrics, slices, and attributions.
(6) **Editability (of $\mathfrak{C}$).** Finally, users would ideally benefit from the ability to modify a released ChangeList to meet their needs e.g. by interactively adding new slices of interest.

How should ChangeLists be created to meet these criteria? What issues arise in their creation? In Section 3.2, we describe an interactive process for API producers to build ChangeLists.

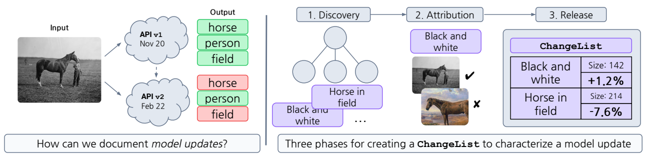


Figure 3: Overview of the process for generating ChangeLists. *(left)* An MLaaS API update changes predictions for downstream users; *(right)* We study the process of building, verifying and releasing ChangeLists. (1) First, we discover slices of where performance has changed, (2) next we ascribe concepts to the slices and verify the attributions with micro-labeling, and (3) we present the slices in an interactive report.

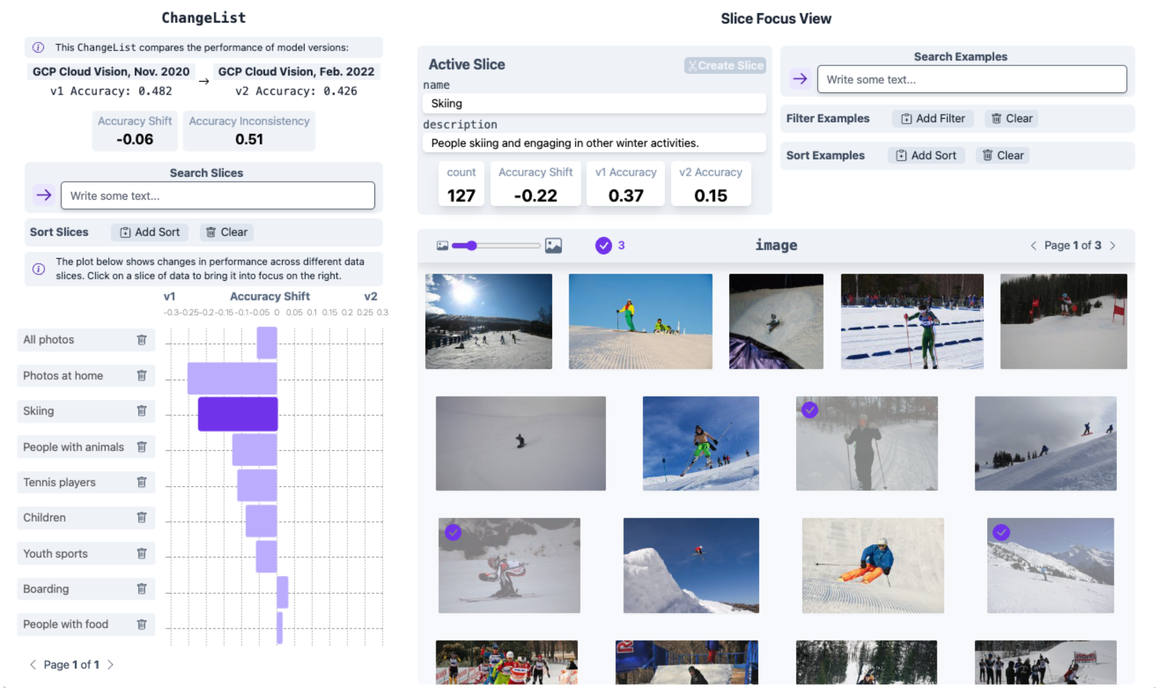
3.2 An Interactive Approach for Generating ChangeLists

We prototype a simple process for building ChangeLists is split into 3 phases:

(1) **Discovery (Section 3.2.1).** First, we identify slices of interest that explain the performance inconsistency σ_ℓ. To discover coherent slices, we adapt the *Domino* [19] *slice discovery* method to our model comparison setting. We also use *Domino* to generate text descriptions for each discovered slice $S^{(j)}$, which serve as initial attributions $A^{(j)}$. We also manually define some slices using interactive tools for search, filtering and labeling, as well as add slices generated by any methods or sources, including programmatically.
(2) **Attribution (Section 3.2.2).** Once slices are discovered, we update the initial slice attributions using interactive slice inspection. A key problem is estimating the alignment of a slice S with its attribution A, while collecting attribute realizations $\{a_i\}_{i=1}^{n}$ on a few informative examples labeled by the user. To address this challenge, we perform *micro-labeling*: an importance sampling procedure driven by CLIP to find the most relevant examples for estimating alignment, coupled with an interface to rapidly label their attribution realizations. Slices with poor attribution alignment can also be updated to improve alignment using a fast training procedure.
(3) **Release (Section 3.2.3).** Finally, once the set of slices (along with their attributions) is finalized, we compile them into a ChangeList. We prototype an *interactive* web application where users can search and sort the ChangeList by the attributions and change metrics. They can also discover and attribute new slices andd add them to the ChangeList. The ChangeList also provides semantic text search over slices, using CLIP to find slices with similar image prototype or text attribution embeddings.

3.2.1 *Discovery.*

Slices are often sourced from metadata or extracted programmatically from the inputs [27]. When working with complex data types (*e.g.* images, natural language), many important slices are not annotated in metadata and cannot easily be extracted programmatically. The limited slices available are insufficient to explain the performance inconsistency, and we must turn to *slice discovery.*



This ChangeList compares the performance of model versions:

GCP Cloud Vision, Nov. 2020 → GCP Cloud Vision, Feb. 2022

v1 Accuracy: 0.482 v2 Accuracy: 0.426

Accuracy Shift	Accuracy Inconsistency
-0.06	0.51

The plot below shows changes in performance across different data slices. Click on a slice of data to bring it into focus on the right.

All photos

Photos at home

Skiing

People with animals

Tennis players

Children

Youth sports

Boarding

People with food

name: Skiing

description: People skiing and engaging in other winter activities.

count	Accuracy Shift	v1 Accuracy	v2 Accuracy
127	-0.22	0.37	0.15

Figure 4: A ChangeList for an image tagging service. (a) *ChangeList View* shows data slices where model performance has improved and degraded. Users can navigate the slices either by issuing search queries or by sorting on size and performance shift. (b) *Slice Focus View* shows examples in the currently selected slice (people skiing and engaging in other winter activities). During the attribution of discovered slices, it also provides rapid labeling tools needed for labeling importance weighted samples (see Section 3.2.2). The implementation of the GUI and back-end is written in Python using a data-wrangling library and is released at: https://github.com/HazyResearch/meerkat.

Slice discovery for model comparison is the task of mining unstructured inputs X for coherent slices that explain the shift inconsistency σ_ℓ. There are many methods that could be used to automatically discover slices [16, 23, 34, 37, 54, 71, 72, 74, 78]. Our objective in this work is not to advocate for one particular slice discovery method, but rather to demonstrate how the discovery phase fits into the process of generating ChangeLists. For readers interested in a comparison of these methods, several works have focused specifically on the problem of benchmarking and evaluating these slice discovery methods [19, 36, 61]. Based on the results from this evaluation, we choose to adapt the *Domino* framework [19] to our new task of explaining model differences in terms of unknown, unlabeled slices. However, we expect many of these other methods could be adapted as well. *Domino* takes as input trained models $v^{[1]}, v^{[2]} : \mathcal{X} \rightarrow \mathcal{Y}$ and a labeled dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n \sim \mathcal{P}(X, Y)$, and outputs slicing functions $\Psi = \{\psi^{(j)} : \mathcal{X} \times \mathcal{Y} \rightarrow [0, 1]\}_{j=1}^k$ that partition the data into k slices $\hat{\mathrm{S}} := \Psi(X, Y) \in [0, 1]^k$. *Domino* proceeds in 3 steps: (1) *embed* the dataset, (2) *slice* the resulting representation space, and (3) *describe* the discovered slices with natural language.

Embed. We embed the dataset $\mathcal{D}$ using an encoder $g_{\text{input}} : \mathcal{X} \rightarrow \mathcal{Z}$ ($\mathcal{Z} \subseteq \mathbb{R}^d$), which yields embeddings $Z = \{z_i := g_{\text{input}}(x_i)\}_{i=1}^n$ for each example. Following Eyuboglu et al. [19], we use CLIP [65] as our encoder when working with images and OpenAI's Ada embeddings when working with text [55].

Slice. We discover slices by fitting a k-component mixture model to the embeddings Z and model losses $\ell^{[1]}, \ell^{[2]}$. For mixture $S^{(j)}$, we assume $Z|S^{(j)}$ varies as multivariate Normal with diagonal covariance. The distribution of losses depends on the loss function ℓ. In the zero-one loss case, we assume $\ell_{01}^{[1]}|S^{(j)}, \ell_{01}^{[2]}|S^{(j)}$ vary as categoricals. We then optimize the log-likelihood with expectation maximization. Like *Domino*, we use a hyperparameter γ to balance the contribution of the embeddings and losses to the log-likelihood – higher γ trades-off coherence for explanatory power.

Describe. Finally, to help users interpret discovered slices, we describe slices in natural language. We source candidate natural language phrases using a large, generative language model. We then identify descriptions $a^{(j)}$ which are closest in embedding space to the centroid of the each slice $S^{(j)}$. For additional details, refer to Eyuboglu et al. [19].

Users can also complement or refine discovered slices using natural language filtering. In the image tagging setting, this is done using inner-product search in CLIP embedding space. In the question answering setting, we feed few-shot prompts to the Flan-T5 open source language model [9]. For details on this approach see Appendix B.1.2.

Once the discovery phase is complete, the ChangeList interface (Fig. 4) displays all discovered and user-specified slices (see Appendix B.3 for manual slicing), along with identified attributions and change metrics.

3.2.2 Attribution. The goal of the attribution phase is to help users verify and edit discovered slices, while communicating the contents of each slice accurately. It proceeds in three steps: (1) *edit* machine attributions by inspecting examples, to align them with the slice; (2) *estimate* alignment between the slice and its attribution; (3) *update* problematic slices that are poorly aligned with their attributions.

Edit Descriptions. When users interpret discovered slices, they typically attribute succinct concepts to slices e.g. "subjects wearing sunglasses" if most images in a slice show a person wearing sunglasses. This attribution allows them to draw conclusions such as "the model update improved by x% accuracy on subjects wearing sunglasses". The prototype interactive components (Fig. 4; discussed in Section 3.2.3) to quickly inspect slices in order to edit machine-generated descriptions. After editing, each slice has a single, textual attribution.

Estimate Alignment. Next, we want to determine the alignment of the slice S with its attribution A using precision P and recall R. Measuring alignment lets users decide if a slice should be kept in the ChangeList, updated to improve alignment, or simply deleted. High precision implies that most slice examples satisfy the attribution i.e. $A = 1$, while with high recall, most examples that satisfy the attribution in the dataset are in the slice. Unfortunately, calculating P, R requires exhaustively labeling the unknown attribute realizations $\{a_i\}_{i=1}^n$ for each example (*i.e.* labeling whether each example satisfies the slice, See 3.1), which is intractable to do for every slice.

Under a small labeling budget, we can only sample a few examples for labeling to estimate $\hat{\text{P}}, \hat{\text{R}}$. While we can estimate $\hat{\text{P}}$ using simple random sampling (see Appendix B.2), naively estimating recall can have high variance [40, 58], since the number of false negatives (examples with $A = 1$ outside the slice) is frequently small relative to the dataset size n.

The key problem is how to construct a *proposal distribution q* that upweights and samples "enough" false negatives to perform estimation via a procedure such as importance sampling [58]. Our insight is to use a cross-modal model like CLIP to construct one or more proposal distributions q_i, by ranking examples in terms of their similarity to the text attribution A. CLIP has the advantage of providing an informative ordering of the examples in response to the wide range of (arbitrarily written) user attributions. A description of our estimation procedure is provided in Appendix B.2. Once the precision and recall are estimated, the user can apply a decision rule (e.g. a minimum threshold on lower confidence bounds) to decide if the slice is satisfactory. If not satisfactory, the user can update it, which we discuss next.

Update Slices. For a slice $S = \psi(X, Y)$ with poor alignment with its attribution A, we can update the slice by training a new slicing function $\tilde{\psi}(X, Y)$, using logistic regression on CLIP embeddings. Ideally, to improve alignment with A, $\tilde{\psi}$ should be aligned with A on the dataset $\mathcal{D}$, i.e. $\tilde{\psi}$ is a good classifier of the attribution realizations $\{a_i\}_{i=1}^n$.

The main challenge is specifying labels for training $\hat{\psi}$, as the $\{a_i\}$ are either unknown, or partially known for previously labeled examples. We use a simple procedure to address this: use attribution labels if available (optionally with additional labeling), otherwise use the original slice labels $\{\psi(x_i, y_i)\}_{i=1}^n$. This updates $\hat{\psi}$ conservatively by matching ψ where necessary, and allows us to systematically improve the slice attribution alignment when a slice is updated. Note that for statistical validity we ensure no overlap between examples used for training or alignment estimation.

3.2.3 Interaction with ChangeLists*.* Finally, we discuss a prototype web interface that allows provide an overview of how producers and users to interact with the ChangeList (Fig. 4). The prototype contains several components which our highlighted in the appendix figures:

(1) **ChangeList View.** (Fig. 8) The left panel ① shows the current ChangeList. The ChangeList is displayed as a barplot against a chosen change metric, with the slice title and size annotated (①c). The user can select any slice for drilldown in the *Slice Focus View*. Users can sort slices in the ChangeList with change metrics (①b), or using semantic search to order slices with the most similar image prototype or slice name embedding first (①a).
(2) **Slice Focus View.** (Fig. 9) The right panel ② displays information about the slice selected in the ChangeList, including tools for performing attribution and navigating the slice. Slice summary statistics are shown along with the ability to edit its name and description (②a). The gallery (②f) enables quick inspection of slice examples, including example selection to display additional metadata. The gallery can be configured to view more or less examples at a glance, and can be sorted and filtered by slice, user and task labels, or any metadata (②d,e). They can also be sorted by semantic similarity to a text search, implemented using CLIP (②c). During attribution, the corresponding component (②b) guides the user through slice updates via training, and attribute quality estimation. The user first passes through an (optional) slice update where the slicing function is retrained using user provided labels. Then, for precision (and recall) estimation, the gallery displays the samples to be labeled for the estimation procedure of Section 3.2.2, and provides keyboard and mouse shortcuts for rapidly selecting and labeling examples. Once calculated, estimates are displayed in the same attribution component.

4 CHANGELISTS ON REAL-WORLD API UPDATES

In this section, we discuss our takeaways from applying the approach described above to six recent API updates and generating a ChangeList for each. First, we provide overview statistics summarizing the changes documented in the ChangeLists. Next, we dive into each API in detail, highlighting noteworthy changes, focusing on those where slice performance goes in the opposite direction as it does globally.

4.1 Task: Image Tagging

Overview of ChangeLists. Using the process described in Section 3, we identified over 113 slices across three real image-tagging API updates. For each discovered slice $S^{(j)}$, we compute the accuracy shift $\mu_{01}^{(j)} \approx \mathbb{E}[D]$ and test the null-hypothesis that the difference in accuracy D is symmetric about zero using the Wilcoxon signed-rank test. On 103 slices, we find that at least one of the API's performance changed significantly (using the Bonferonni correction for multiple hypothesis testing, $\alpha = \frac{0.05}{k}$). Among these, 63 instances of an API's performance *degraded* significantly and on 52, the performance degraded by more than 5%-points. This phenomenon, where an update improves performance globally, but hurts performance on a coherent slice, has not been documented at this scale in the literature. The largest performance shifts are shown in Figure 5. We perform formal attribution for a subset of these discovered slices and provide the estimated recall and precision in Table 4.

In Section 2, we motivate the need for ChangeLists by showing that our updates introduced inconsistency that was not captured by the performance shift. We can quantify how much of the inconsistency is "explained" by our slices with the coefficient of determination r^2. Our ChangeLists achieve quite different r^2 on each update: 16.7% on EveryPixel, 12.4% on Google, and 5.3% on Microsoft. These low r^2 values highlight the difficulty of collecting a comprehensive set of slices and the importance of interactive ChangeLists that allow users to find additional slices. These statistics are summarized in Figure 1.

ChangeList (Google Cloud Vision). Google's API is used in diverse settings ranging from historic photos classification in newspaper archives [30] to managing visual assets in cloud storage [41]. Even though the API improved *on average* after the update, it is important to identify fine-grained slices where performance has degraded. In Figure 5, we show 10 slices where performance degraded. Notably, the API's accuracy in detecting stop signs decreased by over 60%-points, a finding with potential safety implications. Post update, the accuracy of the API's "person" tag drops by 20.8%-points if the person is skiing or snowboarding. If they are playing baseball, accuracy drops by 40.9%-points, and if the photo is in black and white it drops by 17.7%-points. This last slice may be of particular interest to a newspaper using the API on archival photos.

ChangeList (Microsoft Computer Vision). Like Google, Microsoft's API is used in diverse settings and backs mobile applications and intelligent software systems [52]. Across the entire dataset, Microsoft's API improved significantly (+4.0% and +5.9% F1). However, our ChangeList includes 14 slices on which the performance drops significantly. For example, accuracy in tagging "horses" degrades by more than 10%-points in old, black and white photos.

ChangeList (EveryPixel Image Recognition). Unlike the other APIs, the average model performance degraded slightly between updates (−0.5% accuracy and −2.9% F1). Still, we were able to find data slices where the API improved. Notably, after the update, its "cat" detection improved across a broad set of contexts: near windows and doors, in the bathroom, and on or near keyboards. In contrast, the Microsoft API, which improved globally, exhibited significantly degraded performance on "cat" detection after the update.

4.2 Task: Question Answering

Overview of ChangeLists. Using the process described in Section 3, we discover 31 slices across the 4, 470 questions in our QA datasets. For each discovered slice $S^{(j)}$, we compute the accuracy shift $\mu_{F1}^{(j)} \approx \mathbb{E}[D]$ and compute a p-value with the Wilcoxon signed-rank test. Note we do not include formal attribution results for these slices (*i.e.* labeling for recall estimates). Authors manually inspected the slices to check for high precision. On 26 slices, we find that at least one of the API's performance changed significantly (using the Bonferonni correction for multiple hypothesis testing, $\alpha = \frac{0.05}{k}$). There were 11 instances where an API's performance *degraded* and in three, the performance degraded by more than 3%-points. (Note that, due to small sample size, these degradations are not statistically significant. See Figure 1 for 95% confidence intervals.) The largest performance shifts are shown in Figure 2.

ChangeList (OpenAI Text DaVinci). We study two versions of Text DaVinci, OpenAI's largest GPT-3 model available. The main difference between the versions is that version 003 is fine-tuned with reinforcement-learning from human feedback (RLHF), while version 002 is not [8, 57]. The new version outperforms the old by $\mu_{F1} = +0.038$ in mean F1-score. However, this gain is not evenly distributed across data slices. On Geography related questions, there was a large improvement of +0.15 in mean F1-score, while on sports and basketball related questions performance degraded by −0.02 and −0.05, respectively. In the safety-critical setting of Biology, RLHF did not improve performance −0.01.

ChangeList (AI21 Jurassic-1 Grande). Like the Text DaVinci update above, the update to AI-21's Jurassic-1 involved instruction fine-tuning, and it led to significant gains in averperformance $\mu_{F1} = +0.059$. But again there were still slices with degradations, notably on Medicinee-related questions −0.02.

ChangeList (Cohere XLarge). Of the three language model, the update to Cohere's XLarge language model led to a smallest average improvement (+0.023), but it had the highest inconsistency in predictions (0.428). On Geography and "where" questions, performance degraded by −0.01 and −0.04, respectively.

5 DISCUSSING CHANGELISTS WITH API PRODUCERS AND CONSUMERS

To better understand the role ChangeLists could play in practice, we performed a semi-structured user study with MLaaS API producers and consumers, modeled on prior work [12, 63, 69, 70].

5.1 Study Procedures

We recruited a diverse pool of $n = 8$ participants that includes $n = 2$ engineers, $n = 2$ product managers, $n = 2$ data scientists, and $n = 2$ open source developers. They work at organizations ranging from a few employees to over 5,000. Of the participants, $n = 5$ work primarily as *API producers*, that is they develop machine learning models that are released outside of their team (*e.g.* via an API or model hub). These producers work on different parts of the machine learning lifecycle including data collection, model validation and model training. The remaining $n = 3$ participants are *API consumers*, that is they work on software that consumes machine learning predictions via an API or model hub. The participants and their backgrounds are summarized in Table 1. The particpants did not receive compensation for their participation. They were recruited using emails to team leads at organizations producing or consuming MLaaS APIs. We conducted a 45-minute semi-structured interview with each participant, split into two parts. The full script for the

ID	Role	Industry	Org. Size	Consumer	Producer	ML Task
P01	Data Scientist	Cloud Computing	5,000+	✗	✓	Face Recognition
P02	Data Scientist	Cloud Computing	5,000+	✗	✓	Face Recognition
P03	Product Manager	Internet	5,000+	✗	✓	Recommender Systems
P04	Software Engineer	ML Services	10 - 100	✓	✗	Generative Text
P05	Open-source Developer	SaaS	< 10	✓	✗	Text Embeddings
P06	Software Engineer	ML Services	10 - 100	✗	✓	Search
P07	Open-source Developer	SaaS	< 10	✓	✗	Generative Text
P08	Product Manager	Cloud Computing	5,000+	✗	✓	Object Detection

Table 1: Background and experience of study participants. API consumers are professionals who use machine learning model predictions served through an API or model hub. API producers are professionals that develop machine learning models that are publicly released (e.g. via an API or a model hub.)

interview can be found in Appendix A.4. After the interviews, the authors manually grouped and synthesized quotations from participants. These groups are provided in Tables 2 and 3.

(1) Pre-interview. Participants answered a series a questions about their role(s) as producers and/or consumers of MLaaS APIs. Producers answered additional questions about internal processes around model updates and consumers answered additional questions about their experience with model updates.

(2) Think-aloud `ChangeList` demonstration. Participants were informed that they would be shown a demonstration of a prototype `ChangeList` and asked to share their thoughts about it. Participants then followed along in a demonstration of a `ChangeList` conducted by the authors. In the demonstration, we evaluated changes in performance of the Google Cloud Vision API on a subset of the LVIS dataset containing images of people. They were instructed to think-aloud about the challenges `ChangeList` addresses and also highlight limitations with the `ChangeList` or information that is missing from the `ChangeList`. At the end of the demonstration, participants were given time to describe any initial reactions they had to using the `ChangeList`. Then they answered a series of targeted questions.

5.2 Findings

We summarize three key findings that emerged from our interviews, centering on attitudes towards the role of notifications in informing API consumers, the part that `ChangeLists` can play in this process, and the emphasis placed on interactivity and personalization by participants. In Appendix A.1 and Tables 2 and 3, we codify and group additional comments from the participants by theme.

(1) **Existing API update notifications are insufficient.** All the model producers we spoke to said that they typically notify users ($n = 5/5$), but consumers said that they found existing updates vague and lacking in requisite detail. An open-source developer using a language model API for a project complained, *"when [blinded model] was upgraded version X to version Y, I was annoyed that I didn't get the email. Communication about updates could be a bit better...the only guidance they give is vague recommendations."* [P07] This is consistent with comments from API producers: that they primarily aim to highlight new capabilities of models, rarely including detailed metrics or degradations in performance. This is concerning given the degradation in slice performance we observed across the six updates evaluated in Section 4.

Some participants suggested that vague notifications were tolerable, as long as consumers ran evaluations on internal data. However, running these evaluations is costly and there is disagreement on how common internal test sets actually are in practice. One engineer at a small company using large language model APIs explained, *"the ideal thing that we should be doing is benchmarking the tasks we care about and rerun every time the model is updated. That being said, it implies a large cost, especially if our benchmark is large."* [P04] And while some claimed that *"any serious team would be managing their own [evaluation] data"* [P05], others thought that organizations are increasingly building on top of these APIs without internal test sets. "*Our project definitely does not have an internal test set. People are probably overestimating how good the internal test sets are at company at projects.*"[P07] If more organizations and individuals start using APIs without robust evaluation data, the importance of detailed notifications will grow.

(2) **Interactive, slice-based `ChangeLists` communicate missing information.** Producers need to notify a broad range of customers, surfacing changes specific to each, but struggle with the labeling effort required to identify and validate slices. "*If we only had one user, it would be easy because we could talk about the specific changes relevant for them, since we know their use cases. But, we have a wide range of customers. So the challenge is, how do we satisfy the customers broadly?*" [P02] One user described accurate slice attribution as being the main hurdle in providing slices tailored for each user, *"One challenge for identifying performance on a fine-grained slice is, how do you select data that conforms to the concept. Say you have a very specific subclass, for example "woman with gray hair and big eyes."* [P01]

Nearly all participants voiced that the `ChangeList`'s focus on slice-based performance (as opposed to aggregate performance or errors on individual examples) is effective for communicating model differences ($n = 6/8$). *""Makes a lot of sense to report performance on groups of images. If they are individual images – wouldn't make sense to show change in performance.* [P01] A software engineer at a small company said, *"I like this slice based evaluation, so that we get some of the finer-grained differences and I like the interface so we can go in and see what each slice is made of."* [P04]

(3) **Access to `ChangeLists` would influence decision-making in practice.** Both producers and consumers felt that `ChangeLists` would meaningfully change the quality of decision-making in their organizations. Most producers commented that a

ChangeList could facilitate communication between stakeholders at their organization ($n = 4/5$). One highlighted that it would enable communication regardless of technical expertise, facilitating collaboration between data scientists, engineers, and project managers for informed decision-making. *"You can really easily show or make your point without needing to dive into code...then the leadership would be able to look at it and make a plan for action."* [P03]

All participants agree that using a ChangeList could influence them to not use the updated model as a user ($n = 8/8$). In particular, several highlighted that it was specifically the slicing that could influence their decisions, *"Yeah that's possible. If there are certain slices of data that I care about the most that have regressed, I would be worried."* [P02] But, they also highlighted doubts about the willingness of producers to host old API versions. "*Yeah it would [influence me to not update the model]... but I don't know if this would be feasible [for the producer] to offer.*" [P03]

In summary, our interviews confirmed the need for improved documentation around API updates. ChangeLists could help by providing details on fine-grained performance changes. Participants suggested that ChangeLists, were they deployed, would impact day-to-day processes around how people release and use MLaaS APIs. In addition to the findings discussed above, we codified other takeaways in Appendix A.

6 CONCLUSION

In this work, we demonstrate and discuss one approach for producing ChangeLists – detailed reports that highlight changes in performance on fine-grained slices of data. We produce a ChangeList for six updates to popular machine learning APIs and find 63 data slices where the update introduced a statistically significant degradation in performance. These results highlight the importance of including fine-grained reporting alongside model updates. The API consumers we spoke with suggested that this increased transparency in model updates would influence their decisions around updating the model. Our discussion with API producers also highlighted some of the present challenges that limit an organizations ability to release ChangeLists.

There are several limitations with this study, which point to directions for future work. As mentioned by a user study participants, discovering slices is challenging and ChangeLists are only as good as their slices. The 140 slices of data identified in this study only explain a tiny fraction of performance inconsistency observed in the updates. Future work in slice discovery could help bridge this gap. Further, since real-world benchmarks often include predefined slices, real-world ChangeLists should include a mix of discovered slices and predefined slices and benchmarks. However, our work only studied discovered slices. Additionally, our question answering dataset was comparatively smaller and we were limited by the cost of language model inference. Working with API producers to provide low-cost inference for evaluation will be important for future ChangeLists. Finally, in our user study participants did not perform the attribution phase. Future work should further validate this step of the process.

Our work highlights the need for improved transparency in updates to MLaaS APIs. It does so in two ways: (1) documenting real-world instances where a model update degrades performance on an important data slice, but the API producer failed to report it, and (2) discussing the need for improved transparency with API consumers in a think-aloud user study. Towards addressing this need, we demonstrate and analyze one potential approach for improving the transparency of model updates. Our study builds on related work in the literature on transparency of machine learning systems [13, 53, 63]. Given that the use of MLaaS API has grown rapidly in recent years and that model updates are frequent, our work presents an important addition to this line of work on the transparency of deployed machine learning systems.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of NIH under No. U54EB020405 (Mobilize), NSF under Nos. CCF2247015 (Hardware-Aware), CCF1763315 (Beyond Sparsity), CCF1563078 (Volume to Velocity), and 1937301 (RTML); US DEVCOM ARL under Nos. W911NF-23-2-0184 (Long-context) and W911NF-21-2-0251 (Interactive Human-AI Teaming); ONR under Nos. N000142312633 (Deep Signal Processing); Stanford HAI under No. 247183; NXP, Xilinx, LETI-CEA, Intel, IBM, Microsoft, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, Google Cloud, Salesforce, Total, the HAI-GCP Cloud Credits for Research program, the Stanford Data Science Initiative (SDSI), National Science Foundation Graduate Research Fellowship, and members of the Stanford DAWN project: Meta, Google, and VMWare. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of NIH, ONR, or the U.S. Government.

REFERENCES

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.
[2] Monica F Bugallo, Victor Elvira, Luca Martino, David Luengo, Joaquin Miguez, and Petar M Djuric. 2017. Adaptive importance sampling: The past, the present, and the future. *IEEE Signal Processing Magazine* 34, 4 (2017), 60–79.
[3] Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*. PMLR, 77–91.
[4] Lingjiao Chen, Tracy Cai, Matei Zaharia, and James Zou. 2021. Did the Model Change? Efficiently Assessing Machine Learning API Shifts. *arXiv preprint arXiv:2107.14203* (2021).
[5] Lingjiao Chen, Zhihua Jin, Sabri Eyuboglu, Christopher Re, Matei Zaharia, and James Y Zou. [n. d.]. HAPI: A Large-scale Longitudinal Dataset of Commercial ML API Predictions. *Advances in Neural Information Processing Systems Datasets and Benchmarks* ([n. d.]).
[6] Lingjiao Chen, Matei Zaharia, and James Y Zou. 2020. Frugalml: How to use ml prediction apis more accurately and cheaply. *Advances in Neural Information Processing Systems* 33 (2020), 10685–10696.
[7] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. *arXiv preprint arXiv:1808.07036* (2018).
[8] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems* 30 (2017).
[9] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416* (2022).
[10] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty

of natural yes/no questions. *arXiv preprint arXiv:1905.10044* (2019).

[11] Cohere. [n. d.]. Generation Model Card. https://docs.cohere.ai/docs/generation-card.

[12] Anamaria Crisan, Margaret Drouhard, Jesse Vig, and Nazneen Rajani. 2022. Interactive Model Cards: A Human-Centered Approach to Model Documentation. In *2022 ACM Conference on Fairness, Accountability, and Transparency* (Seoul, Republic of Korea) *(FAccT '22)*. Association for Computing Machinery, New York, NY, USA, 427–439. https://doi.org/10.1145/3531146.3533108

[13] Anamaria Crisan, Margaret Drouhard, Jesse Vig, and Nazneen Rajani. 2022. Interactive Model Cards: A Human-Centered Approach to Model Documentation. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency* (<conf-loc>, <city>Seoul</city>, <country>Republic of Korea</country>, </conf-loc>) *(FAccT '22)*. Association for Computing Machinery, New York, NY, USA, 427–439. https://doi.org/10.1145/3531146.3533108

[14] James H Martin Daniel Jurafsky. 2021. Word Senses and WordNet. In *Speech and Language Processing*. 10.

[15] Terrance de Vries, Ishan Misra, Changhan Wang, and Laurens van der Maaten. 2019. Does Object Recognition Work for Everyone?. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[16] Greg d'Eon, Jason d'Eon, James R Wright, and Kevin Leyton-Brown. 2022. The Spotlight: A General Method for Discovering Systematic Errors in Deep Learning Models. *AAAI* (July 2022).

[17] Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

[18] EveryPixel. [n. d.]. Everypixel (EPixel) Image Tagging API. https://labs.everypixel.com/api.

[19] Sabri Eyuboglu, Maya Varma, Khaled Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Ré. 2022. Domino: Discovering Systematic Errors with Cross-Modal Embeddings. In *International Conference on Learning Representations*.

[20] Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

[21] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. 2019. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757* (2019).

[22] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. 2021. Datasheets for datasets. *Commun. ACM* 64, 12 (2021), 86–92.

[23] Robert Geirhos, Kristof Meding, and Felix A Wichmann. 2020. Beyond accuracy: quantifying trial-by-trial behaviour of CNNs and humans by measuring error consistency. *Advances in Neural Information Processing Systems* 33 (2020), 13890–13902.

[24] Robert Geirhos, Kantharaju Narayanappa, Benjamin Mitzkus, Tizian Thieringer, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. 2021. Partial success in closing the gap between human and machine vision. *Advances in Neural Information Processing Systems* 34 (2021).

[25] Thomas Krendl Gilbert, Nathan Lambert, Sarah Dean, Tom Zick, Aaron Snoswell, and Soham Mehta. 2023. Reward reports for reinforcement learning. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*. 84–130.

[26] Karan Goel, Laurel Orr, Nazneen Fatema Rajani, Jesse Vig, and Christopher Ré. 2021. Goodwill hunting: Analyzing and repurposing off-the-shelf named entity linking systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers*. 205–213.

[27] Karan Goel, Nazneen Fatema Rajani, Jesse Vig, Zachary Taschdjian, Mohit Bansal, and Christopher Ré. 2021. Robustness Gym: Unifying the NLP Evaluation Landscape. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*. 42–55.

[28] Raphael Gontijo-Lopes, Yann Dauphin, and Ekin D Cubuk. [n. d.]. No One Representation to Rule Them All: Overlapping Features of Training Methods. *International Conference on Learning Representations* ([n. d.]).

[29] Google. [n. d.]. Google Vision API. https://cloud.google.com/vision.

[30] Sam Greenfield. 2018. Picture what the cloud can do: How the New York Times is using Google Cloud to find untold stories in millions of archived photos. https://cloud.google.com/blog/products/ai-machine-learning/how-the-new-york-times-is-using-google-cloud-to-find-untold-stories-in-millions-of-archived-photos.

[31] Agrim Gupta, Piotr Dollar, and Ross Girshick. 2019. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5356–5364.

[32] Hossein Hosseini, Baicen Xiao, and Radha Poovendran. 2017. Google's Cloud Vision API is Not Robust to Noise. In *16th IEEE International Conference on Machine Learning and Applications, ICMLA 2017, Cancun, Mexico, December 18-21, 2017*, Xuewen Chen, Bo Luo, Feng Luo, Vasile Palade, and M. Arif Wani (Eds.). IEEE, 101–105. https://doi.org/10.1109/ICMLA.2017.0-172

[33] Hossein Hosseini, Baicen Xiao, and Radha Poovendran. 2019. Studying the Live Cross-Platform Circulation of Images With Computer Vision API: An Experiment Based on a Sports Media Event. *International Journal of Communication* 13 (2019), 1825–1845.

[34] Saachi Jain, Hannah Lawrence, Ankur Moitra, and Aleksander Madry. 2022. Distilling model failures as directions in latent space. *arXiv preprint arXiv:2206.14754* (2022).

[35] Hengrui Jia, Hongyu Chen, Jonas Guan, Ali Shahin Shamsabadi, and Nicolas Papernot. 2021. A Zest of LIME: Towards Architecture-Independent Model Distances. In *International Conference on Learning Representations*.

[36] Nari Johnson, Ángel Alexander Cabrera, Gregory Plumb, and Ameet Talwalkar. 2023. Where Does My Model Underperform? A Human Evaluation of Slice Discovery Algorithms. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing* 11, 1 (Nov. 2023), 65–76. https://doi.org/10.1609/hcomp.v11i1.27548

[37] Michael P Kim, Amirata Ghorbani, and James Zou. [n. d.]. Multiaccuracy: Black-box post-processing for fairness in classification. In *AIES 2019*.

[38] Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA Reading Comprehension Challenge. *Transactions of the Association for Computational Linguistics* 6 (2018), 317–328. https://doi.org/10.1162/tacl_a_00023

[39] Allison Koenecke, Andrew Nam, Emily Lake, Joe Nudell, Minnie Quartey, Zion Mengesha, Connor Toups, John R Rickford, Dan Jurafsky, and Sharad Goel. 2020. Racial disparities in automated speech recognition. *Proceedings of the National Academy of Sciences* 117, 14 (2020), 7684–7689.

[40] Jannik Kossen, Sebastian Farquhar, Yarin Gal, and Tom Rainforth. 2021. Active testing: Sample-efficient model evaluation. In *International Conference on Machine Learning*. PMLR, 5753–5763.

[41] Ben Kus. 2017. Box: Bringing image recognition and OCR to cloud content management.

[42] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics* 7 (2019), 453–466.

[43] Yuanchun Li, Ziqi Zhang, Bingyan Liu, Ziyue Yang, and Yunxin Liu. 2021. ModelDiff: testing-based DNN similarity comparison for model reuse detection. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 139–151.

[44] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110* (2022).

[45] Opher Lieber, Or Sharir, Barak Lenz, and Yoav Shoham. 2021. Jurassic-1: Technical details and evaluation. *White Paper. AI21 Labs* 1 (2021).

[46] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V (Lecture Notes in Computer Science, Vol. 8693)*, David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer, 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

[47] Horia Mania, John Miller, Ludwig Schmidt, Moritz Hardt, and Benjamin Recht. 2019. Model similarity mitigates test set overuse. *Advances in Neural Information Processing Systems* 32 (2019).

[48] Neil G Marchant and Benjamin IP Rubinstein. 2021. Needle in a Haystack: Label-Efficient Evaluation under Extreme Class Imbalance. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1180–1190.

[49] GA McIntyre. 1952. A method for unbiased selective sampling, using ranked sets. *Australian journal of agricultural research* 3, 4 (1952), 385–390.

[50] Katelyn Mei, Sonia Fereidooni, and Aylin Caliskan. 2023. Bias Against 93 Stigmatized Groups in Masked Language Models and Downstream Sentiment Classification Tasks. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency* (<conf-loc>, <city>Chicago</city>, <state>IL</state>, <country>USA</country>, </conf-loc>) *(FAccT '23)*. Association for Computing Machinery, New York, NY, USA, 1699–1710. https://doi.org/10.1145/3593013.3594109

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

[52] Microsoft. 2024. Microsoft Release Notes.

[53] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*. 220–229.

[54] Rahul Nair, Massimiliano Mattetti, Elizabeth Daly, Dennis Wei, Oznur Alkan, and Yunfeng Zhang. 2021. What Changed? Interpretable Model Comparison.. In *IJCAI*. 2855–2861.

[55] Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. 2022. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005* (2022).

[56] OpenAI. [n. d.]. How do text-davinci-002 and text-davinci-003 differ? https://help.openai.com/en/articles/6779149-how-do-text-davinci-002-and-text-davinci-003-differ.

[57] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155* (2022).

[58] Art B. Owen. 2013. *Monte Carlo theory, methods and examples.*

[59] Van L Parsons. 2014. Stratified sampling. *Wiley StatsRef: Statistics Reference Online* (2014), 1–11.

[60] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles.* 1–18.

[61] Gregory Plumb, Nari Johnson, Ángel Alexander Cabrera, and Ameet Talwalkar. 2022. Towards a More Rigorous Science of Blindspot Discovery in Image Classification Models. *arXiv preprint arXiv:2207.04104* (2022).

[62] Fait Poms, Vishnu Sarukkai, Ravi Teja Mullapudi, Nimit S Sohoni, William R Mark, Deva Ramanan, and Kayvon Fatahalian. 2021. Low-Shot Validation: Active Importance Sampling for Estimating Classifier Performance on Rare Categories. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 10705–10714.

[63] Mahima Pushkarna, Andrew Zaldivar, and Oddur Kjartansson. 2022. Data Cards: Purposeful and Transparent Dataset Documentation for Responsible AI. In *2022 ACM Conference on Fairness, Accountability, and Transparency* (Seoul, Republic of Korea) *(FAccT '22)*. Association for Computing Machinery, New York, NY, USA, 1776–1826. https://doi.org/10.1145/3531146.3533231

[64] Haode Qi, Lin Pan, Atin Sood, Abhishek Shah, Ladislav Kunc, Mo Yu, and Saloni Potdar. 2020. Benchmarking commercial intent detection services with practice-driven evaluations. *arXiv preprint arXiv:2012.03929* (2020).

[65] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning.* PMLR, 8748–8763.

[66] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).

[67] Arsénio Reis, Dennis Paulino, Vítor Filipe, and João Barroso. 2018. Using Online Artificial Vision Services to Assist the Blind - an Assessment of Microsoft Cognitive Services and Google Cloud Vision. In *Trends and Advances in Information Systems and Technologies - Volume 2 [WorldCIST'18, Naples, Italy, March 27-29, 2018] (Advances in Intelligent Systems and Computing, Vol. 746)*, Álvaro Rocha, Hojjat Adeli, Luís Paulo Reis, and Sandra Costanzo (Eds.). Springer, 174–184. https://doi.org/10.1007/978-3-319-77712-2_17

[68] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. *arXiv preprint arXiv:2005.04118* (2020).

[69] Negar Rostamzadeh, Diana Mincu, Subhrajit Roy, Andrew Smart, Lauren Wilcox, Mahima Pushkarna, Jessica Schrouff, Razvan Amironesei, Nyalleng Moorosi, and Katherine Heller. 2022. Healthsheet: development of a transparency artifact for health datasets. In *2022 ACM Conference on Fairness, Accountability, and Transparency.* 1943–1961.

[70] Hong Shen, Leijie Wang, Wesley H Deng, Ciell Brusse, Ronald Velgersdijk, and Haiyi Zhu. 2022. The model card authoring toolkit: Toward community-centered, deliberation-driven AI design. In *2022 ACM Conference on Fairness, Accountability, and Transparency.* 440–451.

[71] Sahil Singla, Besmira Nushi, Shital Shah, Ece Kamar, and Eric Horvitz. 2021. Understanding failures of deep networks via robust feature extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 12853–12862.

[72] Nimit Sohoni, Jared Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. 2020. No Subclass Left Behind: Fine-Grained Robustness in Coarse-Grained Classification Problems. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 19339–19352. https://proceedings.neurips.cc/paper/2020/file/e0688d13958a19e087e123148555e4b4-Paper.pdf

[73] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615* (2022).

[74] Dennis Wei, Rahul Nair, Amit Dhurandhar, Kush R Varshney, Elizabeth Daly, and Moninder Singh. 2022. On the Safety of Interpretable Machine Learning: A Maximum Deviation Approach. *Advances in Neural Information Processing Systems* 35 (2022), 9866–9880.

[75] Xiaofei Xie, Lei Ma, Haijun Wang, Yuekang Li, Yang Liu, and Xiaohong Li. 2019. DiffChaser: Detecting Disagreements for Deep Neural Networks.. In *IJCAI.* 5772–5778.

[76] Sijie Yan, Yuanjun Xiong, Kaustav Kundu, Shuo Yang, Siqi Deng, Meng Wang, Wei Xia, and Stefano Soatto. 2021. Positive-congruent training: Towards regression-free model updates. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 14299–14308.

[77] Yuanshun Yao, Zhujun Xiao, Bolun Wang, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. 2017. Complexity vs. performance: empirical analysis of machine learning as a service. In *Proceedings of the 2017 Internet Measurement Conference, IMC 2017, London, United Kingdom, November 1-3, 2017*, Steve Uhlig and Olaf Maennel (Eds.). ACM, 384–397. https://doi.org/10.1145/3131365.3131372

[78] Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. 2020. On completeness-aware concept-based explanations in deep neural networks. *Advances in Neural Information Processing Systems* 33 (2020), 20554–20565.

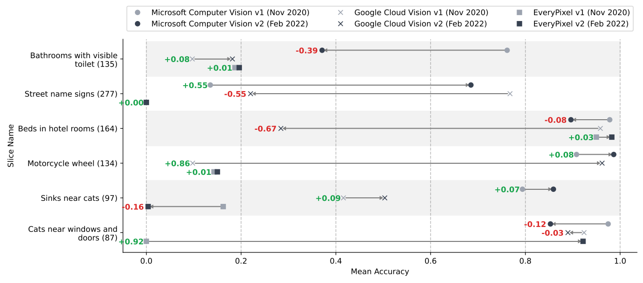


Figure 5: Shifts in Image Tagging Performance on Slices. For six slices, we show the change accuracy for all three question answering APIs. The slices with the largest improvement and degradation for each API are shown. The x-axis shows the accuracy. The y-axis shows the name ascribed to the slice in attribution and its size in parentheses.

A EXTENDED DESCRIPTION OF USER STUDY

A.1 Findings from pre-interviews.

In our interviews with industry professionals, we found that there is mismatch between the kinds of notifications API consumers need and the notifications API producers are currently releasing. In this section, we discuss common themes from our interviews and connect them to our empirical findings from the Section 4. First we discuss responses from model producers, then model consumers. For a breakdown of the participant's comments, see Table 2.

API producers on existing update notifications. All the model producers we spoke to said that they typically notify users ($n = 5/5$), but a couple also suggested that in some circumstances models are changed without notification ($n = 2/5$). *"If it was a small optimization of an internal metric, then there may be less of a notification to the user. But if the model starts working on a new category of data or it has a new capability, then we'll give an update."* [P03] Additionally, when producers do notify users, they primarily aim to highlight new capabilities of models, rarely including detailed metrics or degradations in performance. This is concerning given that in all six evaluated in Section 4 we saw at least one slice of data where model performance degraded.

API producers on what makes notification challenging. Producers need to notify a broad range of customers, surfacing changes specific to each, but struggle with the labeling effort required to identify and validate slices. "*If we only had one user, it would be easy because we could talk about the specific changes relevant for them, since we know their use cases. But, we have a wide range of customers. So the challenge is, how do we satisfy the customers broadly?*"[P02] One user described accurate slice attribution as being the main hurdle in providing slices tailored for each user, *"One challenge for identifying performance on a fine-grained slice is, how do you select data that conforms to the concept. Say you have a very specific subclass, for example "woman with gray hair and big eyes."* [P01]

API producers on the importance of notifications. Producers recognize the importance of detailed notifications for customers' applications. All of the producer we interviewed said that notifications were important for ($n = 5/5$). *"Its super important. There's a real need, and the notification actually needs to depend on the customer. We don't know what the customer does with API downstream."* [P02]

API consumers on existing update notifications. Though producers typically notify users, consumers find that existing updates are vague and lack requisite detail. An open-source developer using a language model API for a project complained, *"when [blinded model] was upgraded version X to version Y, I was annoyed that I didn't get the email. Communication about updates could be a bit better...the only guidance they give is vague recommendations."* [P07] This is consistent with what API producers explained: that they rarely include metrics or degradations.

API consumers on the ideal update notification. Consumers who apply APIs in narrow use cases want fine-grained metrics in updates from producers. *"If there were better metrics that could say this model does this and this model does this. Some way of evaluating how well the model does on your distribution and on your data... consumer reports for language models."*[P07] Though some that use APIs in more general settings said that they were okay trusting the general improvements reported by the producer.

API consumers on internal evaluations. Several mentioned that running evaluations on internal data is important, but running these evaluations is costly and there is disagreement on how common internal test sets actually are in practice. One engineer at a small company using large language model APIs explained, "the ideal thing that we should be doing is benchmarking the tasks we care about and rerun every time the model is updated. That being said, it implies a large cost, especially if our benchmark is large." [P04] And while some claimed that *"any erious team owuld be managing their own [evaluation] data"* [P05], others think that folks are increasingly building on top of these APIs without internal test sets. "*Our project definitely does not have an internal test set. People are probably overestimating how good the internal test sets are at company at projects.*"[P07]

A.2 Findings from think-aloud demonstration.

Through their comments during our think-aloud demonstration, participants expressed that the slice-based design of `ChangeLists` make them useful for understanding differences between models and give them the potential to influence producer and consumer decision-making. In this section, we discuss common themes from the comments and feedback provided by participants. to our empirical findings from the Section 4. First we discuss responses from model producers, then model consumers. For a breakdown of the participant's comments, see Table 2.

On the utility of the `ChangeLists` for understanding model updates. Participants expressed interest in using the ChangeList, though some voiced that its utility hinges on the quality of the groupings and the context in which its used. Without being explicitly prompted to say so, over half of the participants ($n = 5/8$) said they would want to use `ChangeLists` to understand differences between models. A product manager who works on data sourcing for ML APIs at a cloud provider said, *"First impression, this is really awesome if I can use it ... [coming from the perspective] of doing data work."* [P08] A software engineer at a small company said, *"I'm impressed...I like this slice based evaluation, so that we get some of* *the finer-grained differences and I like the interface so we can go in and see what each slice is made of."*[P04] Others expressed interest, but said its ultimate utility would depend on the quality of the slices, *"This is really interesting. But it really depends on how good the automatic grouping is."*[P07]

On the importance of the slicing functionality. Nearly all participants voiced that reporting slice performance (as opposed to aggregate performance or errors on individual examples) is effective for communicating model differences ($n = 6/8$). *""Makes a lot of sense to report performance on groups of images. If they are individual images – wouldn't make sense to show change in performance.*[P01]

On the importance of personalization. Participants commented on the importance of personalizing the `ChangeList`, highlighting the value of a data upload feature in the `ChangeList` ($n = 7/8$). *"Every company has their own definitional data, its critical to upload your own data."* [P08] In addition, several participants suggested a "playground" feature where users could craft new inputs to the model and explore differences in the predictions.

On potential of `ChangeLists` to influence producer decision-making. Most of the producers commented that the tool could facilitate communication between stakeholders at their organization ($n = 4/5$). One highlighted that it enables communication regardless of scientific background, allowing for easy presentation of data and facilitating collaboration between data scientists, engineers, and project managers for informed decision-making. *"You can really easy show or make your point without needing to dive into code...Then the leadership would be able to look at it and make a plan for action."*[P03]

On the potential of `ChangeLists` to influence consumer decision-making. Participants agree that using a ChangeList could influence them to not use the updated model as a user ($n = 8/8$). In particular, several highlighted that it was specifically the slicing that could influence their decisions, *"Yeah that's possible. If there are certain slices of data that I care about the most that have regressed, I would be worried."*[P02] But, they also highlight doubts about the willingness of producers to host old API versions. "*Yeah it would [influence me to not update the model]... but I don't know if this would be feasible [for the producer] to offer.*"[P03]

A.3 Study Procedures

Recruitment. The API producers were recruited by reaching out to relevant teams at cloud providers and smaller companies. The API consumers were recruited by reaching out to small companies and leaders of open-source projects that use ML APIs.

Transcription. If participants agreed, the interview was recorded, transcribed and de-identified. Otherwise, the interviewer noted their answers by hand. After the session, the notes were edited for clarity and sent back to the participant for review.

A.4 Interview Script

Thank you for participating in our user study on model updates.

This is a think-aloud user study. You will be asked to use a software tool and share your thoughts and experiences while using it. During the session, we will record your interactions with the tool as well as your verbal and written comments. The interviews will be conducted over Zoom and recorded using Zoom software. The recordings will be transcribed and de-identified, then discarded after the session. Your comments may be used in qualitative analyses and de-identified quotations from your comments may be included in any resulting publications.

A.4.1 Part 1: Pre-interview. **Question:** In your work, do you develop machine learning models that are publicly released (*e.g.* via an API or a model hub)? This includes data collection, model validation, model training, hyperparameter tuning?

If yes,

- **Question:** What kind of models do you deploy (*i.e.* what is the input modality, the prediction target, the intended use case)?
- **Question:** Who are the intended users of your models?
- **Question:** How frequently do you update the models (*e.g.* retrain, change the architecture, recalibrate, change the output schema)?
- **Question:** Do you notify users when you update a model?
- **Question:** What information do you include in the update?
- **Question:** What challenges do you encounter when communicating model updates to users?
- **Question:** Do you think users should be notified when the model is updated?
- **Question:** Ignoring the feasibility of collecting the data, what information do you think users should have after a model update?
- **Question:** What software packages or tools do you use to understand changes in model performance between updates?
- **Question:** Do you report changes in fine-grained slices? If so, how do you find them. What challenges are associated with this.

Question: In your work, do you use machine learning model predictions served through an API or model hub (*e.g.* GPT-3, Google Cloud Vision, HuggingFace, TorchVision)?

If yes,

- **Question:** What models and services do you use in your work and what role do they play?
- **Question:** Are you made aware of updates to the APIs you are using? If so, how does the provider make you aware?
- **Question:** When the model is updated, what information would be important for you to be made aware of?
- **Question:** Has a model update ever had adverse affects on your system or use case?
- **Question:** How did you measure the degradation?
- **Question:** Outside of the information given to you by the provider, what tools (if any) do you use to understand changes in model performance between updates?
- Question: Would you contribute to a public, crowdsourced changelist for a model that you are using?

A.4.2 Part 2: `ChangeList` *Demonstration.* We'll be demonstrating a `ChangeList`, a prototype of an interactive tool we've developed that helps users analyze changes in model performance. Before, during and after using the tool we'll ask you some questions.

Instruction. As we use the tool, please "think-aloud" about:

(1) What parts of the `ChangeList` are easy to use/intuitive?

MLaaS API Producers		
Synthesis	**Quote**	**Participant**
Topic: On existing update notifications. *Synthesis: Producers aim to highlight new capabilities of models, but rarely include metrics or regressions.*	"If it was a small optimization of an internal metric, then there may be less of a notification to the user. But if the model starts working on a new category of data or it has a new capability, then we'll give an update."	P03
	"For example, maybe previously the model could only do indoor images, but now we can support outdoor performance as well. On the other hand, if there certain distributions that we haven't looked into, we don't report those."	P02
	"If they are paying for the service, there may be an NDA version of the update that will include more detailed metrics. But that's not going to be released publicly."	P03
Topic: On what makes notification challenging. *Synthesis: Producers need to notify a broad range of customers, surfacing changes specific to each, but struggle with the labeling effort required to identify and validate subgroups.*	"One challenge for identifying performance on a fine-grained slice is, how do you select data that conforms to the concept. Say you have a very specific subclass, for example 'woman with gray hair and big eyes'. So this would be more difficult than, say, the coarse gained demographic slices."	P01
	"Here we can use existing labels or dispatch a labeling effort. So its going to depend a bit on how good the human annotation team is."	P03
	"If we only had one user, it would be easy because we could talk about the specific changes relevant for them, since we know their use cases. But, we have a wide range of customers. So the challenge is, how do we satisfy the customers broadly?"	P02
Topic: On the importance of notifications. *Synthesis: Producers recognize the importance of detailed notifications for customers' applications.*	"Its super important. There's a real need, and the notification actually needs to depend on the customer. We don't know what the customer does with API downstream."	P02
	"Do you think users should be notified when the model is updated? In this case, absolutely."	P03
MLaaS API Consumers		
Synthesis	**Quote**	**Participant**
Topic: On existing update notifications. *Synthesis: Consumers agree that updates are vague and lack detail.*	"We don't really know what's updated under the hood. [It] just says that its better in practice. We don't have a sense of what has changed."	P04
	"The update was pretty vague. There was some box on the website that said ... our embeddings stuff has changed....there was certainly no multi-dimensional analysis of where it was better and where it was worse."	P05
	"When [blinded model] was upgraded version 2 to version 3, I was annoyed that I didn't get the email. Communication about updates could be a bit better...The only guidance they give is vague recommendations."	P07
Topic: On the ideal update notification. *Synthesis: Consumers who use APIs in narrow use cases want fine-grained metrics in updates from producers.*	"If there were better metrics that could say this model does this and this model does this. Some way of evaluating how well the model does on your distribution and on your data... consumer reports for language models."	P07
	"If I were relying on a very specific case of the model, then you could imagine a case where the model performs a lot worse on one particular case. But since I was using it in a more general setting, I trust that the general improvement is representative of what I'm going to see."	P04
Topic: On internal evaluations. *Synthesis: Several mentioned that running evaluations on internal is important, but running these evaluations is costly and there is disagreement on how common internal test sets actually are in practice.*	"Ideal thing that we should be doing is benchmarking the tasks we care about and rerun every time the model is updated. That being said, it implies a large cost, especially if our benchmark is large."	P04
	"Probably, any serious team would be managing their own data. But there are certainly a lot of young projects being built that don't have rigorous validation."	P05
	"Our project definitely does not have an internal test set. People are probably overestimating how good the internal test sets are at company at projects, but this is just a gut feeling."	P07

Table 2: Quotes from pre-interviews with study participants. Participant comments were manually grouped by topic. Then each group of comments was synthesized for common themes.

(2) What parts of the ChangeList are difficult to use/unintuitive?
(3) What challenges does ChangeList address?
(4) What challenges does ChangeList not address?

- **Demo.** Show the discover button and a set of previously discovered slices. Explain that each bar corresponds to a slice of data and the change in performance on that slice.
- **Demo.** Click on one of the slices. Show how you can explore examples of that slice on the right.
- **Demo.** Name the slice and provide a description. Use the search functionality to ensure that the name is representative

Synthesis	Quote	Participant
Topic: On the utility of the ChangeList for understanding model updates. *Synthesis: Participants expressed interest in using the ChangeList, with some voicing that its utility hinges on the quality of the groupings and the context in which its used.*	"This interface is really cool. I would definitely use it."	P01
	"First impression this is really awesome ... [coming from the perspective] of doing data work."	P08
	"I'm impressed...I like this slice based evaluation, so that we get some of the finer-grained differences and I like the interface so we can go in and see what each slice is made of."	P04
	"How helpful this would be would depend on the skillset of the user. For someone who is familiar with the task and looking at the data on regular basis, I can see this being helpful. But for someone who is just introduced to the task and not looking at the data, I don't think they can make a conclusion."	P02
	"This is really interesting. But it really depends on how good the automatic grouping is. "	P07
	"I can absolutely imagine using it to compare different APIs. I less so imagine using it if the model API changed. In most cases the simplest thing to do is to try it out on internal and then just go back to the other one."	P05
Topic: On the importance of the slicing functionality. *Synthesis: Participants agree that reporting subgroup performance is important for identifying and communicating regressions.*	"With a tool like this, it would be very easy for the customer to prove that the subgroup that they are interested in has regressed.	P03
	"I think the table on the left is pretty clear [in informing the differences between the models]. It calls out which groups show the biggest variance"	P08
	"Makes a lot of sense to report performance on groups of images. If they are individual images – wouldnt make sense to show change in performance."	P01
	"The nice thing with this tool is its like, here are the groups we do well with and here are the groups that we don't do well on."	P03
Topic: On personalization. *Synthesis: Participants highlighted the importance of users being able to upload their own data to the ChangeList.*	"It would be really really helpful if it was contextualized within the previous queries. I would like to be able to look at my previous data."	P06
	"Every company has their own definitional data, its critical to upload your own data."	P08
Topic: On communication between stakeholders. Synthesis: Participants agree that the tool effectively facilitates communication between stakeholders, regardless of scientific background.	"The nice thing about a tool like this is it bridges everybody without regard for scientific background. You can really easy show or make your point without needing to dive into code...Then the leadership would be able to look at it and make a plan for action."	P03
	"It would definitely be useful...its healthy to have the data scientists and engineers work alongside the PMs."	P02
Topic: On the potential impact of ChangeLists. *Synthesis: Participants note that using a ChangeList could influence them to not use the updated model as a user.*	"Yeah it would [influence me to not update the model]... but I don't know if this would be feasible to offer."	P03
	"Yeah I think ideally yes I would want to stick with the old model. But in reality, idk how often it can be done, it may not make sense for the company delivering the API."	P04
Topic: On areas for improvement. *Synthesis: Participants made suggestions for improvements in layout and design.*	"First impression is there are a lot of buttons."	P08
	"I think i'm definitely more informed. But, I also feel overwhelemed by so many pictures. I think some sort of summarization would be useful."	P01

Table 3: Quotes from think-aloud `ChangeList` demonstration. Participant comments were manually grouped by topic. Then each group of comments was synthesized for common themes.

of the slice. Explain how we also support attribution, which uses statistical methods to rapidly estimate the validity of a name.

- **Demo.** Search for slices that are relevant to the user using the slice search.
- **Demo.** Suppose we have some internal data that is specific to some internal application. Show that we can upload that data to the CL using the upload button.
- **Question.** Do you have any initial reactions to the tool?
- **Question.** After exploring the `ChangeList`, do you feel more informed about the differences between the two models? Please explain.
- **Question.** As a model user, do you feel that a `ChangeList` could influence you to not update the model, and continue using the old version?

- **Question.** As a model user, would you interact with a `ChangeList` before updating a model?
- **Question.** As a model user, how important is the feature that allows users to upload their own data? Would you use it? Do you expect others would?
- **Question.** As a model user, how important is the feature that allows users to discover and identify new slices, and share them with the community?
- **Question.** What role could ChangeLists play in your work and at your organization? Would this be valuable tool for sharing updates with stakeholders in the organization?
- **Question.** What other information would you have liked to have seen?

B EXTENDED DESCRIPTION OF METHODOLOGY

B.1 Discovery

B.1.1 Discovery with Domino. In this section, we provide details on the slice discovery techniques used in this work. In general, we follow closely the approach [19].

We discover slices by fitting a k-component mixture model to the embeddings Z and model losses $\ell^{[1]}, \ell^{[2]}$. For mixture $S^{(j)}$, we assume $Z|S^{(j)}$ varies as multivariate Normal with diagonal covariance. The distribution of losses depends on the loss function ℓ. In the zero-one loss case, we assume $\ell_{01}^{[1]}|S^{(j)}, \ell_{01}^{[2]}|S^{(j)}$ vary as categoricals. The log-likelihood over the validation dataset is given as follows and mazimied using expectation-maximization:

$$\ell = \sum_{i=1}^{n} \log \sum_{j=1}^{\bar{k}} P(S^{(j)}=1) P(Z=z_i|S^{(j)}=1) P(\ell^{[1]}=y_i|S^{(j)}=1)^{\gamma} P(\ell^{[2]}|S^{(j)}=1)^{\gamma}, \tag{3}$$

Like *Domino*, we use a hyperparameter γ to balance the contribution of the embeddings and losses to the log-likelihood – higher γ trades-off coherence for explanatory power. A slice is *coherent* if the examples in it share something common. A set of slices have explanatory power if membership in those slices can explain the shift inconsistency.

B.1.2 Slice Discovery with Natural Language Filters. When working with the Question Answering datasets, the slices discovered by Domino were useful for slice ideation, but were poorly aligned with attributed concepts. So, we used Domino to seed ideas for slices and then used a few-shot prompting strategy with Flan T5 to more accurately filter the dataset.

For example, to filter the dataset down to the slice containing questions about biology, we used the following prompt:

```
Are the sentences below about biology?
"What was the city of beijing previously known as?"
Answer: Not Biology

"Has a country won the world cup at home?"
Answer: Not Biology

"Where is fe best absorbed in the body?"
Answer: Biology

"What is the function of the pericardial sac?"
Answer: Biology

"Where are antibodies made and by what type of lymphocyte?"
Answer: Biology

"{question}"
Answer:
```

We ran this prompt through the language model (with temperature 0.1) for each question in the dataset substituting it in the template. The dataset was filtered to the set the questions for which the model output "Biology".

These slices can be verified in the attribution phase just like the ones discovered with Domino alone.

B.2 Attribution: Alignment Estimation

We would like to estimate the precision and recall with only a small amount of labeling effort:

$$\mathrm{P} = \frac{\sum_{i=1}^{n} s_i a_i}{\sum_{i=1}^{n} s_i} \qquad \mathrm{R} = \frac{\sum_{i=1}^{n} s_i a_i}{\sum_{i=1}^{n} a_i}$$

Consider one of the slices S that was discovered in the first phase of our process (Section 3.2.1). Because $S = \psi(X, Y)$, we can compute the realizations of the slice variable $\{s_i = \psi(x_i, y_i)\}_{i=1}^{n}$ across our full dataset. On the other hand, we cannot access any of the realizations of the attributions $\{a_i\}$, since they are unknown.

Estimating Precision. We estimate precision directly using the standard approach of Monte Carlo estimation with simple random sampling (SRS). To estimate precision $\hat{\mathrm{P}}$, we first sample n_{P} examples to label with their attribution realizations $\{a_i\}$, and then compute the estimator $\hat{\mathrm{P}} = \frac{\sum_{i=1}^{n_{\mathrm{P}}} 1[a_i=1]}{n_{\mathrm{P}}}$. We use a standard bootstrap procedure to compute a confidence interval around the estimated precision [17].

Estimating Recall. Unfortunately, estimating recall efficiently is difficult since the number of false negatives (i.e. examples with $A = 1$ that lie outside the slice) can be small relative to the size of the dataset, making SRS an inefficient method with high variance in this setting. Beyond SRS, there are many approaches to sampling and estimation with a small sample size including stratified sampling [59], importance sampling [58], ranked set sampling [49] and others, as well as adaptive variants [2]. Estimating recall with limited labels has also recently received more attention in the machine learning community, particularly with adaptive approaches [40, 48, 62].

Among these approaches, importance sampling is a strong and reliable baseline, and we leave the exploration of adaptive methods to future work. For simplicity, we reduce recall estimation to a two step process: (1) using *mixture* importance sampling [58] to estimate the proportion Q of examples with the attribution $A = 1$ in the complement of the slice; and (2) using a plug-in estimator for recall with the estimates for precision $\hat{\mathrm{P}}$ and proportion $\hat{\mathrm{Q}}$. Formally, we define Q,

$$\mathrm{Q} = \frac{\sum_{i=1}^{n} (1 - s_i) a_i}{\sum_{i=1}^{n} (1 - s_i)}$$

In the first step, we use mixture importance sampling i.e. a simple variant of the Horvitz–Thompson estimator that is unbiased [58]. Key to this method is the choice of the proposal distributions q_i, which upweight samples that are likely to be useful for estimation [58]. Indeed, the key problem is how to construct *proposal* *distributions* that sample "enough" false negatives for estimation via importance sampling [58]. Good choices for the q_i (i.e. those that lead to low variance estimates) would put higher weight on the less prevalent samples with $A = 1$, and lower weight on those with $A = 0$. At first glance, this appears impossible without labeling the attribute realizations $\{a_i\}$. While prior work has studied the estimation of classifier recall with limited labels [40, 48, 62], these all reuse the classifier being evaluated to construct a proposal distribution. We do not have a classifier for arbitrary (user constructed) text attributions A in our setting.

Instead, we propose a procedure that relies on a flexible method to construct proposal distributions. Our insight is to use CLIP (or any cross-modal foundation model) to construct one or more proposal distributions q_i, by ranking examples in terms of their similarity to the text attribution A. The advantage of using CLIP in this way is that it can provide an informative ordering of the examples in response to the wide range of (arbitrarily written) user attributions. This in turn leads to proposal distributions that are more likely to appropriately upweight samples that correspond to the concept A, which may have been arbitrarily selected by the user.

In detail, each example x_j in the population is assigned a score λ_{ij} based on inner-product search with respect to text queries $i \in [d]$ written by the user. Here, the user will write text queries that they think align with the attribution A. The similarity score λ_{ij} serves as a useful proxy for whether the example satisfies the attribution A, and the ranking of examples by λ_{ij} should correlate with the attribution realizations. Then, we construct a proposal distribution q_i from each set of scores by first min-max scaling the scores, and then powering them in order to skew the distribution i.e. $q_i(x_j) \propto \left(\frac{(\lambda_{ij} - \min_k \lambda_{ik})}{(\max_k \lambda_{ik} - \min_k \lambda_{ik})}\right)^r$ for an exponent r. This serves to create a proposal distribution that assigns very low probability to examples that have the lowest scores.

Once the proposal distributions q_i are created, we construct a mixture distribution $q_\alpha = \sum_i \alpha_i q_i$ with $\sum_i \alpha_i = 1$ (by default, we use the uniform mixture $\alpha_i = \frac{1}{d}$). We sample n_{Q} examples from the mixture distribution with corresponding weights w_j (with $w_j = \sum_i \alpha_i w_{ij}$) and user provided attribution realizations a_j. We can then estimate the proportion of examples $\hat{\mathrm{Q}}$ in the slice complement with $A = 1$, as well as the recall $\hat{\mathrm{R}}$ using an (unbiased) plug-in estimator,

$$\hat{\mathrm{Q}} = \frac{\sum_{j=1}^{n_{\mathrm{Q}}} \frac{1[a_j=1]}{w_j} \cdot \frac{1}{n - \sum_{i=1}^{n} s_i}}{n_{\mathrm{Q}}}, \qquad \hat{\mathrm{R}} = \frac{(n - n_s) \cdot \hat{\mathrm{P}}}{(n - n_s) \cdot \hat{\mathrm{P}} + n \cdot \hat{\mathrm{Q}}}$$

We provide confidence intervals for recall by running a standard bootstrap procedure independently for both $\hat{\mathrm{P}}$ and $\hat{\mathrm{Q}}$, and combine these independent estimates to get bootstrapped estimates for recall. We then output the appropriate quantiles corresponding to the required confidence level.

Once the precision and recall are estimated, the user can apply a consistent decision rule (e.g. a minimum threshold on lower confidence bounds) in order to decide if the slice is satisfactory. If not satisfactory, the user can update it, as discussed in Section 3.2.2.

B.3 Manually Gathering Slices

Manually gathering slices is a critical process for refining outputs of slice discovery methods (SDMs) and for creating slices that were not automatically discovered. However, a manual step requires scalable data exploration, which is difficult to do with large datasets. In the `ChangeList` prototype, users can rapidly scrub through data in the gallery and label examples to assign them to the appropriate slice (Fig. 10). Users can also create their own slices and label examples that are part of that slice.

We also leverage image-text foundation models, like CLIP, to perform similarity search between image examples and text queries (Fig. 11). Similarity search can reduce the burden of having to scrub through large datasets when the attributes of interest are not labeled. Similarity searches can also be used to find semantically meaningful groups of images, which can expedite manual slice discovery workflows.

C EXTENDED DESCRIPTION OF THE LONGITUDINAL DATABASE OF API PREDICTIONS

C.1 Task: Image Tagging

In image tagging, the input X is an image and category (*e.g. horse*) pair and the target $Y \in \{0, 1\}$ is a binary label indicating whether an object in the category is in the image. We consider the point-wise zero-one loss $\ell(y, \hat{y}^{[i]}) = \mathbf{1}[y = \hat{y}^{[i]}]$. We also report other metrics that are not point-wise: recall, precision, and F1-score.

Dataset. We use the Large Vocabulary Instance Segmentation (LVIS) dataset, a relabeling of the original Common Objects in Context (COCO) images [31, 46]. The dataset has $n = 1{,}577{,}603$ examples. LVIS labels have two advantages over the original COCO labels. (1) LVIS includes over 1,203 categories (compared to the 80 in COCO), which better reflects the breadth of categories output by modern image tagging APIs. (2) LVIS provides *negative sets*, a set of images for each category where **no** instance of the category appears. This allows us to measure both the precision and F1-score of the APIs, while still using a long-tail set of categories.

APIs. We consider three object detection APIs: Google's AutoML Vision Object Detection API [29], EveryPixel's Image Keywording Service [18], and Microsoft Computer Vision Image Understanding API [51]. The predictions are sourced from *History of APIs* (HAPI), a longitudinal database of API predictions [5]. We use the raw outputs of the APIs and perform our own preprocessing that maps the labels output by the APIs to those in LVIS (see Section C for details).

Reconciling labels. The label set output by image recognition APIs will not necessarily match that of the evaluation dataset. For example, LVIS includes labels for 1,723 different object categories, while the 2020 version of the Google API output over 7,462 different object categories [31]. In order to evaluate an API's performance on a dataset, we must first reconcile the two category sets. If an API outputs a category not in the LVIS vocabulary (*e.g.* "toboggan"), we want to map it to a more general category in the LVIS vocabulary (*e.g.* "sled"). To do so, we leverage the WordNet lexical database [20], collecting for each category in LVIS all words with a more

name	count	recall	precision
Snowboard airborne	257	0.971 (0.94, 1.00)	0.995 (0.98, 1.00)
Stop signs	103	0.631 (0.54, 0.76)	0.947 (0.89, 0.99)
Street name signs	277	0.789 (0.73, 0.85)	0.986 (0.97, 1.00)
Cats in the bathroom	119	0.889 (0.83, 0.95)	1.000 (1.00, 1.00)
Dogs in the house	561	0.801 (0.75, 0.85)	0.980 (0.97, 0.99)
Horses in old photos	101	0.974 (0.92, 1.00)	1.000 (1.00, 1.00)
Horses in rural settings	446	0.913 (0.87, 0.96)	0.930 (0.91, 0.96)
Surfboards on lakes and rivers	35	1.000 (1.00, 1.00)	1.000 (1.00, 1.00)
Surfboards in the ocean	614	0.859 (0.82, 0.92)	1.000 (1.00, 1.00)
Surfboards away from water	173	0.959 (0.89, 1.00)	1.000 (1.00, 1.00)
Motorcycle wheel	134	0.795 (0.73, 0.86)	1.000 (1.00, 1.00)
Train and bus wheel	349	0.936 (0.89, 0.98)	1.000 (1.00, 1.00)
Airplane wheel	187	0.981 (0.96, 1.00)	1.000 (1.00, 1.00)
Skateboard wheel	186	0.663 (0.62, 0.72)	1.000 (1.00, 1.00)
Skiing person	135	0.957 (0.91, 0.99)	0.981 (0.95, 1.00)

Table 4: Estimates of precision and recall for measuring attribution alignment with slices found across three image tagging API updates. Slices such as "stop signs" and "skateboard wheel" have low recall, so they may be rejected for inclusion in a final `ChangeList`, while all other slices have both precision and recall above 0.7.

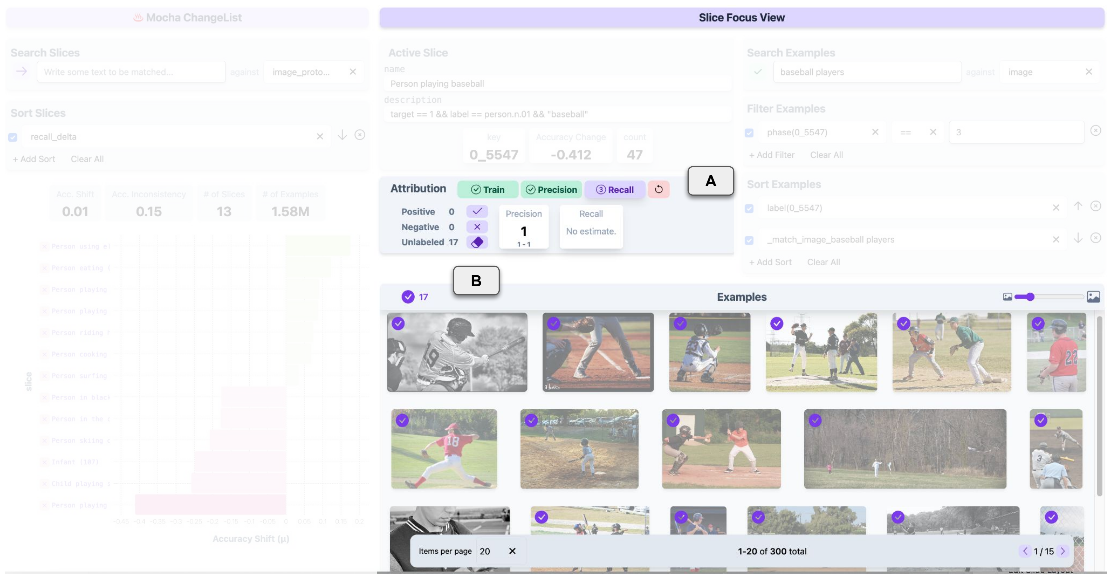


Figure 6: Slice Refinement and Coherence Statistics. The attribution panel (A) provides an fast zero-one data labeling interface, which allows users to efficiently refine slices, bootstrap refiner models, and compute coherence metrics (e.g. precision, recall) on the chosen slice. Users batch select examples in the gallery (B) and select one of three options to label: 1) positive, 2) negative, 3) unlabeled (i.e. erase).

specific meaning (*i.e.* its hyponyms). We find the hyponyms of a category using the following procedure:

(1) For each **category** in evaluation dataset, get the corresponding WordNet **synsets**. (LVIS categories are already based on WordNet synsets.)
(2) For each **synset** compute all (direct and indirect) **hyponyms**.
(3) For each **hyponym** collect all of its **lemmas** and filter them to down only include those whose most common noun word sense (based on WordNet sense ordering, see [14]) is the hyponym. This gives us a mapping from each lemma to a list of categories.
   - *Note*: This serves to filter out improbable mappings. For example, the synset "mouse.n.02" is defined as "a person who is quiet or timid" and is a hyponym of "person". However,

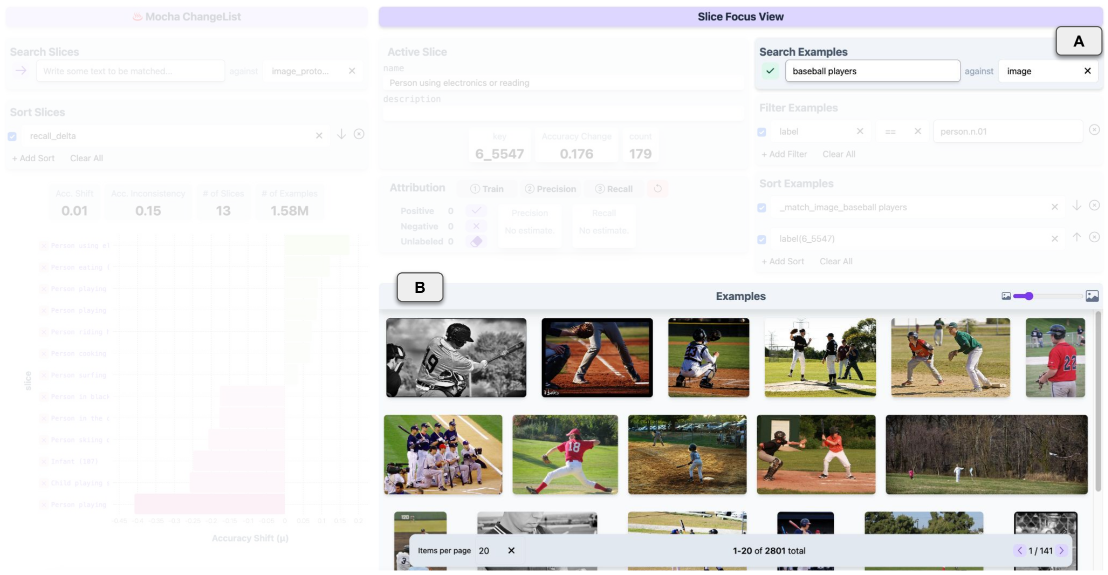


Figure 7: Searching for Examples. Our `ChangeList` prototype supports semantic similarity search between images and unstructured text using image-text foundation models, like CLIP. Based on the user search query (A), images in the selected slice (or, if no slice is selected, entire dataset) are sorted by their similarity to the query (B).

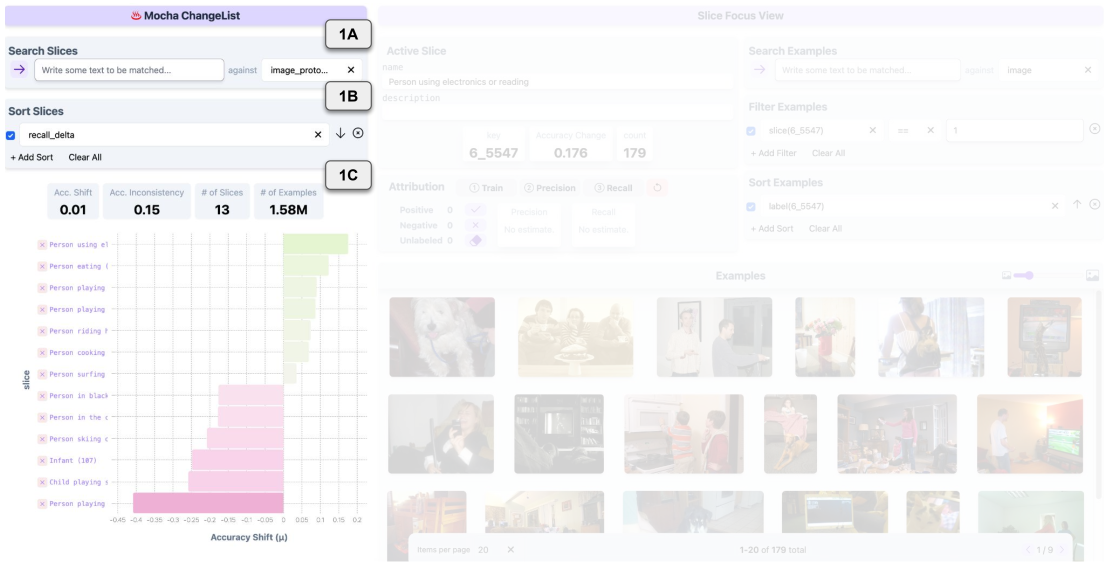
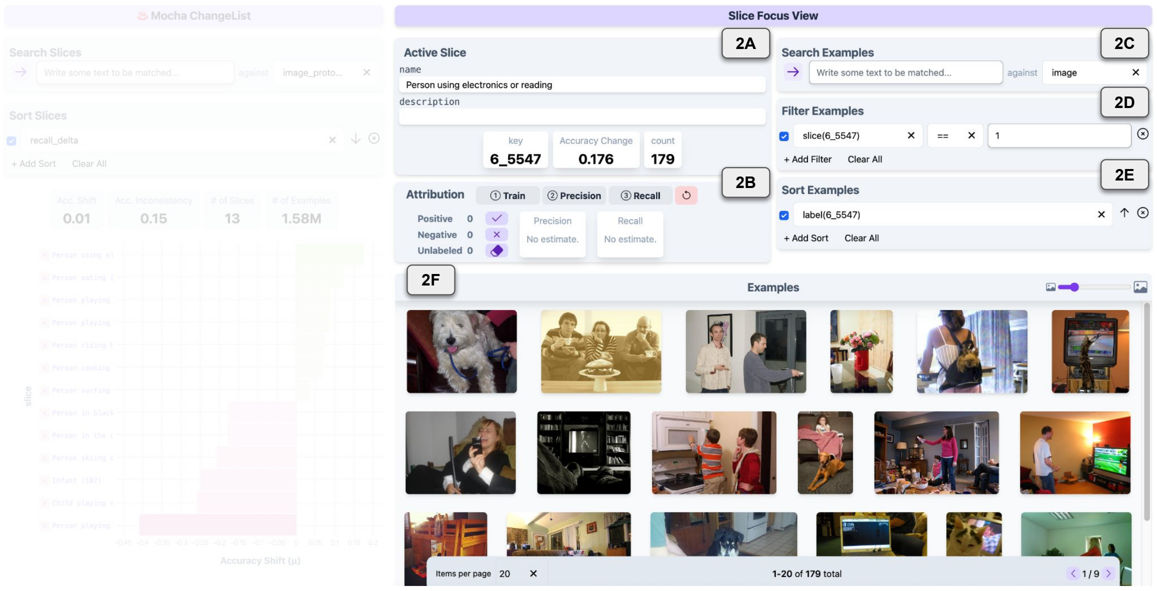


Figure 8: `ChangeList` View. The `ChangeList` panel consolidates information about the current `ChangeList`. Users can search for slices using text-based semantic queries, which match slices with the most similar image prototype or slice name (component 1A). Slices can also be ordered by associated metadata, such as change in performance or number of examples in the slice (component 1B). The barplot summarizes changes in a user-selected metric across the different slices (component 1C).

this leads to an odd mapping: a prediction of "mouse" is mapped to the category "person". To avoid this, we only consider the most common noun word sense of a lemma, which for "mouse" describes a rodent.

(4) For each **lemma** select the category with the highest path similarity between its synset and the lemma's hyponym. This gives us a mapping from each lemma to a single category.

Using the resulting mapping, we can then "translate" the set of categories predicted by the API to the set of categories used in the dataset. If a predicted category is not in the mapping, we ignore it.



name: Person using electronics or reading
description:

key	Accuracy Change	count
6_5547	0.176	179

Positive 0
Negative 0
Unlabeled 0

Precision: No estimate.
Recall: No estimate.

Search Examples: Write some text to be matched... against image

Filter Examples: slice(6_5547) == 1

Sort Examples: label(6_5547)

Search Slices: Write some text to be matched... against image_proto...

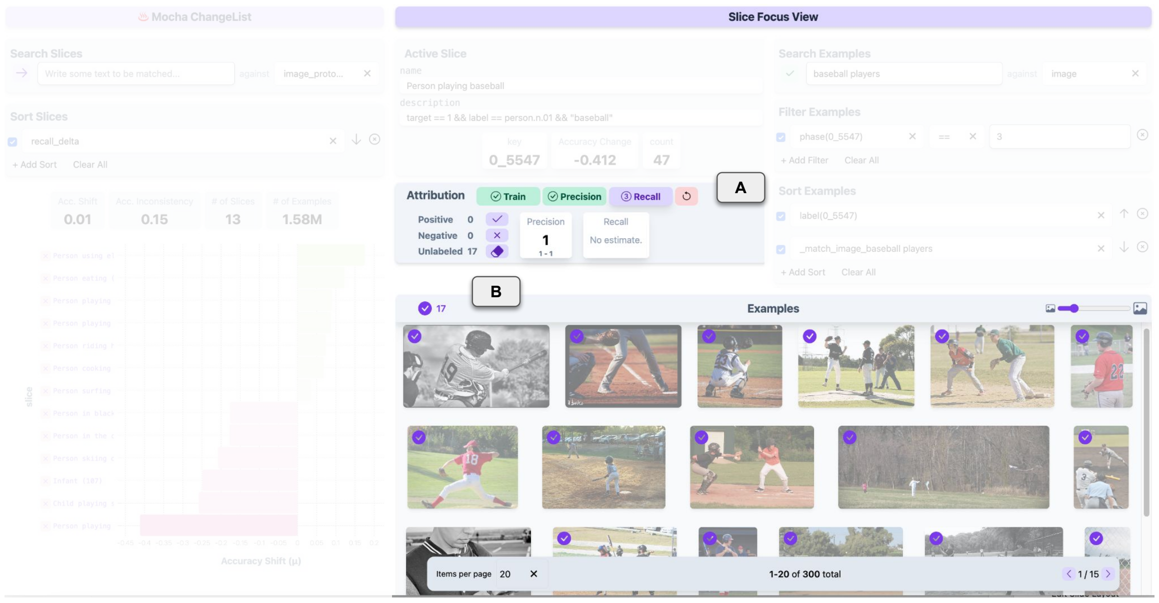
Sort Slices: recall_delta

Acc. Shift	Acc. Inconsistency	# of Slices	# of Examples
0.01	0.15	13	1.58M

Items per page 20 1-20 of 179 total 1 / 9

Figure 9: Slice Focus View. The slice focus vie enables granular inspection of different examples in the slice. Users can visualize characteristics of the selected (active) slice (component 2A) and manually label different attributes in the dataset (component 2B). Users can also search for examples that match unstructured text queries (component 2C) and filter and sort examples by existing or generated metadata (components 2D,2E). All examples are ordered in the gallery, which enables efficient data scrubbing.

This means that each predicted category is mapped to *at most one* dataset category. If the object belongs in both specific and more general categories (*e.g.* "canoe" and "boat") the API is expected to output both the specific and general categories. This is based on the recommended evaluation approach provided by the LVIS authors [31].



Write some text to be matched... against image_proto...

recall_delta

+ Add Sort Clear All

Acc. Shift	Acc. Inconsistency	# of Slices	# of Examples
0.01	0.15	13	1.58M

name: Person playing baseball

description: target == 1 && label == person.n.01 && "baseball"

key	Accuracy Change	count
0_5547	-0.412	47

baseball players against image

phase(0_5547) == 3

label(0_5547)

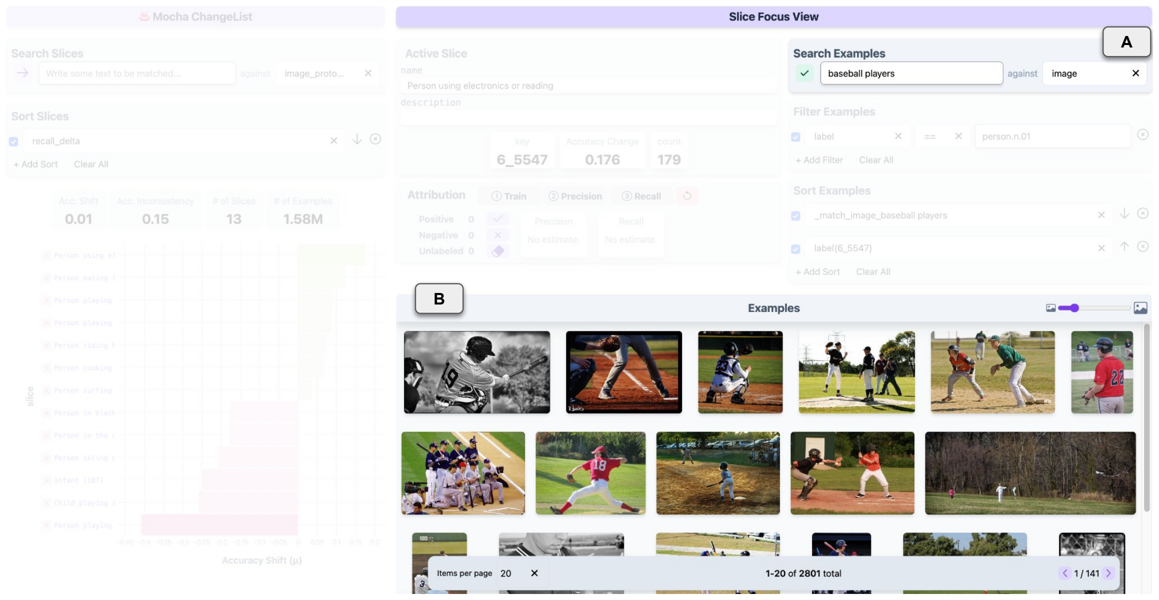
_match_image_baseball players

		Precision	Recall
Positive	0	1	No estimate.
Negative	0	1 - 1	
Unlabeled	17		

17

Items per page 20 1-20 of 300 total 1 / 15

Figure 10: Slice Refinement and Coherence Statistics. The attribution panel (A) provides an fast zero-one data labeling interface, which allows users to efficiently refine slices, bootstrap refiner models, and compute coherence metrics (e.g. precision, recall) on the chosen slice. Users batch select examples in the gallery (B) and select one of three options to label: 1) positive, 2) negative, 3) unlabeled (i.e. erase).



name: Person using electronics or reading

key	Accuracy Change	count
6_5547	0.176	179

baseball players against image

label == person.n.01

_match_image_baseball players

label(6_5547)

Items per page 20 1-20 of 2801 total 1 / 141

Figure 11: Searching for Examples. The `ChangeList` supports semantic similarity search between images and unstructured text using image-text foundation models, like CLIP. Based on the user search query (A), images in the selected slice (or, if no slice is selected, entire dataset) are sorted by their similarity to the query (B).