Agile Modeling: From Concept to Classifier in Minutes

Otilia Stretcu* [1], Edward Vendrow* [1,2], Kenji Hata* [1], Krishnamurthy Viswanathan[1], Vittorio Ferrari[1], Sasan Tavakkol[1], Wenlei Zhou[1], Aditya Avinash[1], Enming Luo[1], Neil Gordon Alldrin[1], MohammadHossein Bateni[1], Gabriel Berger[1], Andrew Bunner[1], Chun-Ta Lu[1], Javier Rey[1], Giulia DeSalvo[1], Ranjay Krishna[3], Ariel Fuxman[1]

[1] Google Research, [2] Stanford University, [3] University of Washington

Contact: otiliastr@google.com, afuxman@google.com

Abstract

The application of computer vision methods to nuanced, subjective concepts is growing. While crowdsourcing has served the vision community well for most objective tasks (such as labeling a "zebra"), it now falters on tasks where there is substantial subjectivity in the concept (such as identifying "gourmet tuna"). However, empowering any user to develop a classifier for their concept is technically difficult: users are neither machine learning experts nor have the patience to label thousands of examples. In reaction, we introduce the problem of Agile Modeling: the process of turning any subjective visual concept into a computer vision model through real-time user-in-the-loop interactions. We instantiate an Agile Modeling prototype for image classification and show through a user study (N=14) that users can create classifiers with minimal effort in under 30 minutes. We compare this user driven process with the traditional crowdsourcing paradigm and find that the crowd's notion often differs from that of the user's, especially as the concepts become more subjective. Finally, we scale our experiments with simulations of users training classifiers for ImageNet21k categories to further demonstrate the efficacy of the approach.

1. Introduction

Whose voices, and therefore, whose labels should an image classifier learn from? In computer vision today, the answer to this question is often left implicit in the data collection process. Concepts are defined by researchers before curating a dataset [13]. Decisions for which images constitute positive versus negative instances are conducted by majority vote of crowd workers annotating this pre-defined set of categories [32, 57]. An algorithm then trains on this aggregated ground truth, learning to predict labels that represent the crowd's majoritarian consensus.

Figure 1: Visual concepts can be nuanced and subjective, differing from how a majoritarian crowd might label a concept. For example, a graduate student may think that well-prepared tuna sandwiches are considered `gourmet tuna`, but sushi chef might disagree.

As computer vision matures, its application to nuanced, subjective use cases is burgeoning. While crowdsourcing has served the vision community well on many objective tasks (e.g., identifying ImageNet [13] concepts like "zebra", "tiger"), it now falters on tasks where there is substantial subjectivity [21]. Everyday people want to scale their own decision-making on concepts others may find difficult to emulate—for example, in Figure 1, a sushi chef might covet a classifier to source gourmet tuna for inspiration. Majority vote by crowd workers may not converge to the same definition of what makes a tuna dish gourmet.

This paper highlights the need for user-centric approaches to developing real-world classifiers for these subjective concepts. To define this problem space, we recognize the following challenges. First, concepts are subjective, requiring users to be embedded in the data curation process. Second, users are usually not machine learning experts; we need interactive systems that elicit the subjective decision boundary from the user. Third, users don't have the patience nor resources to sift through the thousands of training instances that is typical for most image classification datasets [13, 35, 29]—for example, ImageNet anno-

*Equal contribution.

tated over 160M images to arrive at their final 14M version.

In order to tackle these challenges, we introduce the problem of **Agile Modeling**: the process of turning any visual concept into a computer vision model through a real-time user-in-the-loop process. Just as software engineering matured from prescribed procedure to "agile" software packages augmenting millions of people to become software engineers, Agile Modeling aims to empower anyone to create personal, subjective vision models. It formalizes the process by which a user can initialize and interactively guide the training process while minimizing the time and effort required to obtain a model. With the emergent few-shot learning capabilities of vision foundation models [46, 24], now is the right time to begin formalizing and developing Agile Modeling systems.

We instantiate an Agile Modeling prototype for image classification to highlight the importance of involving the user-in-the-loop when developing subjective classifiers. Our prototype allows users to bootstrap the learning process with a single language description of their concept (e.g., "gourmet tuna") by leveraging vision-language foundation models [46, 24]. Next, our prototype uses active learning to identify instances which, if labeled, would maximally improve classifier performance. These few instances are surfaced to the user, who is only asked to identify which instances are positive—something they can do even without a background in machine learning. This iterative process continues with more active learning steps until the user is satisfied with their classifier's performance.

Our contributions are:

1. We formulate the Agile Modeling problem, which puts users at the center of the image classification process.
2. We demonstrate that a real-time prototype can be built by leveraging SOTA image-text co-embeddings for fast image retrieval and model training. With our optimizations, each round of active learning operates over over 10M images and can be performed on a single desktop CPU in a few minutes. In under 5 minutes, user-created models outperform zero-shot classifiers.
3. In a setting resembling real-world conditions, we compare models trained with labels from real users versus crowd raters, and find that the value of user-labeled data increases when the concept is nuanced or difficult.
4. We verify the results of the user study with a simulated experiment of 100 more concepts in ImageNet21k.
5. We open source the implementation of our Agile Modeling prototype on our GitHub page [59], enabling anyone to create classifiers for their concepts.
6. We release all annotations labeled in our user study for 14 novel concepts, enabling researchers to experiment with the concepts defined by our users [59].

2. Related work

Our work draws inspiration from human-in-the-loop, personalization, few-shot, and active learning.

Building models with humans-in-the-loop. Involving humans in the training process has a long history in crowdsourcing [15, 1, 42, 17], developmental robotics [60, 28, 36], and computer vision [31, 11, 68, 30, 41]; and has recently also grown in popularity in large language modeling [40]. However, these methods are primarily focused on improving model behavior. In other words, they ask "how can we leverage human feedback or interactions to make a better model?" In comparison, we take a user-centric approach and ask "how can we design a system that can empower users to develop models that reflect their needs?"

With this framing in mind, our closest related work comes from the systems community [43, 47, 64, 39]. Tropel [43] automated the process of large-scale annotation by having users provide a single positive example, and asking the crowd to determine whether other images are similar to it. Nevertheless, for subjective concepts—particularly those with multiple visual modes—a single image may be insufficient to convey the meaning of the concept to the crowd. Others such as Snorkel [47, 64] circumvented large-scale crowd labeling through the use of expert-designed labeling functions to automatically annotate a large, unlabeled dataset. However, in computer vision, large datasets of images contain metadata that is independent of the semantics captured within the photo [61]. With the recent emergent few-shot capabilities in large vision models, it is now time to tackle the human-in-the-loop challenges through a modeling lens appropriate for the computer vision community. Our prototype can train a model using active learning on millions of images on a single CPU in a matter of minutes.

Perhaps the closest work to ours is [39], which proposes a method of interleaving model training and labeling phases to build classifiers for rare concepts. However, our work differs in several ways: (1) our method obviates the requirement of having a few positive images by allowing users to find them quickly with natural language, (2) we use concepts proposed by real users, rather than using concepts from existing benchmarks, and (3) we demonstrate that our system works in real-time on a much larger scale of data: our method can train a model and run active learning on millions of images on a single CPU in a few minutes, which is much faster than [39]'s end-to-end training of a ResNet.

Personalization in computer vision. Although personalization [26, 7, 20] is an existing topic in building classification, detection, and image synthesis, the settings being tackled are different than ours. For example, in [7] personalized concepts refer to objective instance-specific concepts (e.g. "my dog"), and the user must provide a few images to begin. [20] tackles the problem of personalized text-to-

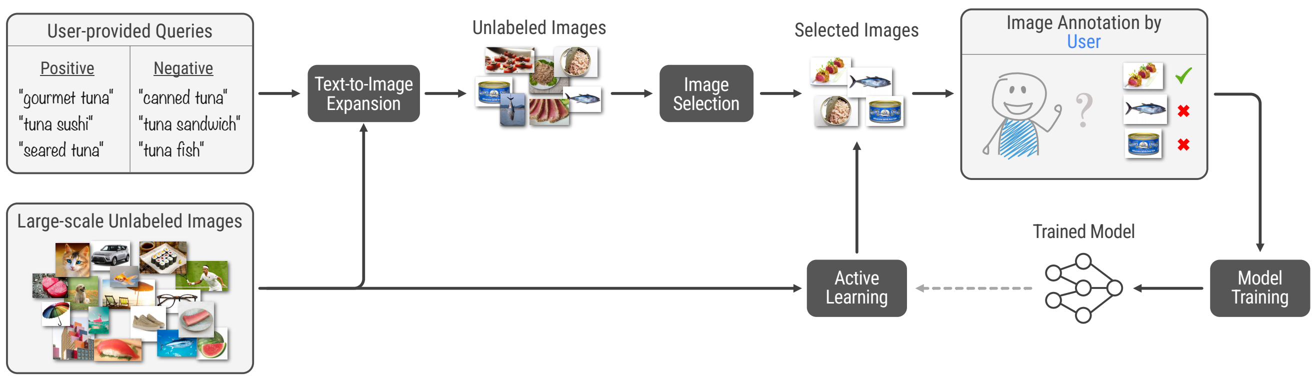


Figure 2: Overview of the Agile Modeling framework. Starting with a concept in the mind of the user, the system guides the user into first defining the concept through a few text phrases, automatically expands these to a set of images, followed by one or more rounds of real-time active learning on a large corpus, where the user only needs to rate images.

image generation. [26] assumes the training images are either given in one go (few-shot) or in a continual learning fashion, and their method has no control over data selection. Most importantly, existing work in this area usually tests their resultant models on standard vision datasets and does not build real-time systems that can enable the user to select the few-shots and later improve the model with active learning, while we run a study with real users, focus on real-world sized datasets and on new, subjective concepts.

Zero and few-shot learning. Since users have a limited patience for labeling, Agile Modeling aims to minimize the amount of labeling required, opting for few-shot solutions [63, 65, 56, 4, 38]. Luckily, with the recent few-shot properties in vision-language models—found for example in CLIP [46] and ALIGN [24]—it is now possible to bootstrap classifiers with language descriptions [45]. Besides functioning as a baseline, good representations have shown to similarly bootstrap active learning [62]. We demonstrate that a few minutes of annotation by users can lead to sizeable gains over these zero-shot classifiers.

Real-time active learning. Usually few-shot learning can only get you so far, especially for subjective concepts where a single language description or a single prototype is unlikely to capture the variance in the concept. Therefore, iterative approaches like active learning provide an appropriate formalism to maximize information about the concept while minimizing the amount of labels needed [55, 5]. Active learning methods derive their name by "actively" asking users to annotate data which the model currently finds most uncertain [33] or believes is most representative of the unlabeled set [54] or both [2, 6]. Unfortunately, most of these methods require expensive pre-processing, reducing their utility in most real-world applications [12]. Methods to speed up active learning limit the search for informative data points [8] or use low-performing proxy models for data selection [9] or use heuristics [54, 44]. We show that performing model updates and ranking images on cached co-embedding features is a scalable and effective way to conduct active learning.

3. Agile Modeling

We consider the scenario where a user comes to the Agile Modeling system with just a subjective concept in mind—in our running example, `gourmet tuna`. First we lay out the high level Agile Modeling problem framework, and then describe how we instantiate a prototype of this framework.

3.1. The framework

As shown in Figure 2, the Agile Modeling framework guides the user through the creation of an image classifier through the following steps:

1. Concept definition. The user describes the concept using text phrases. They are allowed to specify both positive phrases, which can describe the concept as a whole or specific visual modes, as well as negative phrases, which are related but not necessarily part of the concept (*e.g.*, `canned tuna` is not gourmet).

2. Text-to-image expansion and image selection. The text phrases are used to mine relevant images from a large unlabeled dataset of images for the user to rate.

3. Rating. The user rates these images through a rating tool, specifying whether each image is either `positive` or `negative` for the concept of interest.

4. Model training. The rated images are used to train a binary classifier for the concept. This is handled automatically by the system.

5. Active learning. The initial model can be improved very quickly via one or more rounds of active learning. This consists of 3 repeated steps: (1) the framework invokes an algorithm to select from millions of unlabeled images to rate; (2) the user rates these images; (3) the system retrains the classifier with all the available labeled data. The whole active learning procedure operates on millions of images and returns a new model in under 3 minutes (measured in Section 4.3.1).

The user's input is used for only two types of tasks, which require no machine learning or engineering experience: first in providing the text phrases and second in rating images. Everything else, including data selection and model training, is performed automatically. With such an automated process, users do not need to hire an machine learning or computer vision engineer to build their models.

3.2. The prototype

We focus our prototype on the core north star task of image classification [16]. One of the main challenges of Agile Modeling is to enable the user to effectively transfer their subjective interpretation of a concept into an operational machine learning model. For our image classification task, Agile Modeling seeks to turn this arbitrary concept into a well-curated training dataset of images. We assume that that the user only has access to a large, unlabeled dataset, which is something that is easily available through the Internet [46]. Our aim is to select and label a small subset of this large dataset and use it as training data.

Concept definition. Users initiate the Agile Modeling process by expressing their concept in words. For example, the user might come in and simply say `gourmet tuna`. However, users can also preemptively provide more than a single phrase. They can also produce negative descriptions of what their concept is *not*. They can clarify that `canned tuna` is not gourmet. Through our interactions with users, we find that expressing the concept in terms of both positive and negative phrases is an effective way of mining positive and hard negative examples for training. The positive phrases allow the user to express both the concept as a whole (*e.g.*, `gourmet tuna`) and specific visual modes of it (*e.g.*, `seared tuna`, `tuna sushi`). The negative phrases are important in finding negative examples that could be easily confused by the model or by raters.

Text-to-image expansion and image selection. The phrases provided by the user are used to identify a first set of relevant training images. To achieve this, we take advantage of recent, powerful image-text models, such as CLIP [46] and ALIGN [24]. We co-embed both the unlabeled image dataset and the text phrases provided by the user into the same space, and perform a nearest-neighbors search to retrieve 100 images nearest to each text embedding. We use an existing nearest-neighbors implementation [67, 22] that is extremely fast due to its hybrid solution of trees and quantization. From the set of all nearest neighbors, we randomly sample 100 images for the user to rate. We do this for both positive and negative phrases, since the negative texts are helpful in identifying hard negative examples.

Data labeling by user. The selected images are shown to the user for labeling. In our experiments, we created a simple user interface where the user is shown one image at a time and is asked to select whether it is `positive` or `negative`. The median time it took to rate a single image by the participants in our user study was 1.7 ± 0.5 seconds (details in Section 4). Since users rate 100 images per annotation round, they spend approximately 3 minutes before a new model is trained.

Model training. We train our binary image classifier using all previously labeled data. This setup is challenging because there is little data available to train a generalizable model, and the entire training process must be fast to enable real-time engagement with the user waiting for the next phase of images. The lack of large-scale data suggests the use of few-shot techniques created to tackle low data scenarios, such as meta-learning [66, 23, 18] or prototype methods [58], however most such approaches are too slow for a real-time user interaction. While the study of real-time few-shot methods is an interesting problem for future instantiations of the Agile Modeling framework, we adopted another solution that helps us address both challenges: we again take advantage of powerful pretrained models like CLIP and ALIGN to train a small multilayer perceptron (MLP), with only 1-3 layers, on top of image embeddings provided by such large pretrained models. These embeddings bring much needed external information to address the low data challenge while allowing us to train a low-capacity model that can be trained fast and is less prone to overfitting. Model architectures and training details are described in Section 4.

Active learning (AL). We improve the classifier in the traditional model-based active learning fashion: (1) we use the current model to run inference on a large unlabeled pool of data, (2) we carefully select a batch of images that should be useful in improving the model, (3) we ask the user to rate these images, (4) we retrain the model. This process can be repeated one or more times to iteratively improve performance. When selecting samples to rate, state-of-the-art AL methods generally optimize for improving the model fastest [48]. However, when the user is the rater, we have a real-time constraint to minimize the user-perceived latency. Therefore, AL methods that rely on heavy optimization strategies cannot be used here. In our solution, we adopt a well-known and fast method called *uncertainty sampling* or *margin sampling* [10, 51, 34], which selects im-

Step	Time
User rates 100 images	2 min 49 sec $\pm$ 58 sec
AL on 10M images	58.6 sec $\pm$ 0.8 sec
Training a new model	23.1 sec $\pm$ 0.2 sec

Table 1: The average and standard deviation of the time it takes per step in our Agile Modeling instantiation. Rating time was measured by taking the average median time of an user to rate one image during the experiments used in this paper. To measure time for AL and model training, they were each run 10 times.

ages for which the model is *uncertain*. Specifically, given a model with parameters θ and a sample x, we define the uncertainty score as $P_\theta(\hat{y}_1|x) - P_\theta(\hat{y}_2|x)$, where $\hat{y}_1$ and $\hat{y}_2$ are the highest and second-highest probabilities predicted by the model. Note that there are other definitions of uncertainty such as least confidence and entropy, but since we are in a binary classification setting, all of these definitions are mathematically equivalent. We also considered the approach adopted by [39], which is a combination of margin and positive mining. In each AL round, the rating phase is divided into sub-rounds, where in each sub-round samples are selected based on one of two strategies: if so far we have rated more positives than negatives, we select samples from the margin, otherwise we pick samples with the highest prediction scores (to mine for positives). We compared both margin and this approach in our experiments. We run one or more rounds of AL, the number of rounds is determined by the time the user has.

We release a Colab implementation of this prototype at our GitHub page [59].

4. Experiments with real users

We run user studies with real users in the loop, and show that: **(1)** In only 5 minutes, the performance of an Agile model can exceed that of state-of-the-art zero-shot models based on CLIP and ALIGN by at least 3% AUC PR (Section 4.3.1); **(2)** For hard, nuanced concepts, Agile models trained with user annotations outperform those trained with crowd annotations even when crowd raters annotate $5\times$ more data (Section 4.3.2); **(3)** Smaller active learning batch sizes perform better than larger ones, but there is an efficiency trade-off (Section 4.4); **(4)** Agile models using ALIGN embeddings outperform does using CLIP throughout model iterations (Section 4.4).

4.1. Choosing subjective concepts

Concepts. For our user studies we select a list of 14 novel concepts, spanning different degrees of ambiguity and difficulty. These concepts were curated by surveying real-world practitioners for suggestions and later filtered to a list of 14 concepts that had multiple subjective interpretations [21]. The list ranges from more objective concepts such as `pie chart`, `in-ear headphones` or `single sneaker on white background`, to more subjective ones such as `gourmet tuna`, `healthy dish`, or `home fragrance`. We found that our concepts cover a large spread over the visual space—we measure this spread using the average pairwise cosine distance between the concept text embeddings (using CLIP). For our 14 concepts, the average pairwise cosine distance was 0.73 ± 0.13. In comparison, ImageNet's average pairwise cosine distance was 0.35 ± 0.11. The full list of concepts is included in Appendix A, along with the queries provided by the users.

Workflow. We provide users with only the concept name and a brief description, but allow them to define the full interpretation. For instance, one of our users, who was provided with the concept `stop-sign`, limited its interpretation to only real-world stop-signs: only stop signs in traffic were considered positive, while stop-sign drawings, stickers, or posters were considered negative[1].

Participants. When collecting data for the experiments, we sourced 14 volunteer users to interact with our system. Each participant built a different concept. None of the users performed any machine learning engineering tasks. Our experiments indicate that it takes participants 2 minutes and 49 seconds on average to label 100 images, as shown in Table 1. Our participants were adults that spanned a variety of age ranges (18-54), gender identities (male, female), and ethnicities (White, Asian, and Middle Eastern).

Data sources. Since our prototype requires an unlabeled source of images from which to source training labels, we use the LAION-400M dataset [53], due to its large size and comprehensive construction based on the large Common Crawl web corpus. We throw away the text associated with the images. We remove duplicate URLs and split the images into a 100M training and 100M testing images. All trained Agile models use data exclusively from the unlabeled training split, including during nearest neighbor search, active learning, and training. For evaluation, we only use data from the 100M test set, where each concept's evaluation set consists of a subset of this data rated by the user.

4.2. Experimental setup

Models and training. All models are multilayer perceptrons (MLP) that take image representations from a frozen pretrained model as input and contain one or more hidden layers. For the first active learning step, we use a smaller MLP with 1 hidden layer of 16 units to prevent overfitting, while all active learning rounds and final model have 3 hidden layers of size 128. All models are trained using binary

[1]This definition was inspired by a self-driving car application, where a car should only react to real stop signs, not those on posters or ads.

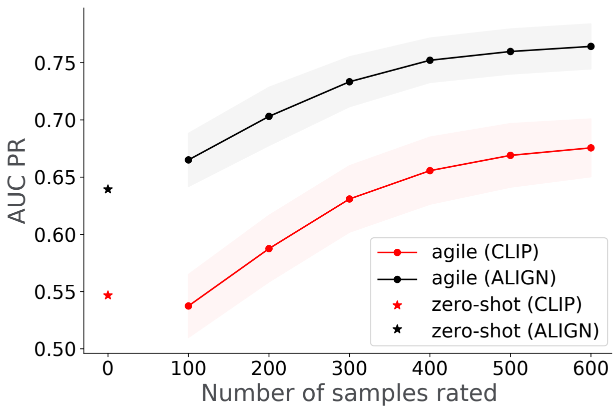


Figure 3: Model performance per amount of samples rated by the user (AUC PR mean and standard error over all concepts). Each ● corresponds to an active learning round.

cross-entropy loss, a dropout rate of 0.5 and weight decay regularization with weight 10^{-4}. We use the Adam optimizer [27] with learning rate 10^{-4} and train for 10 epochs. To prevent overtriggering by the trained classifier, we sample 500k random images from the unlabeled set and automatically label them negative. During training, we upsample our labeled positives to be half the training set, while labeled negatives and the random negatives are each a quarter of the training set. All hyperparameters have been chosen on 2 held-out concepts.

Baselines. One baseline we compare against is zero-shot learning, which corresponds to zero effort from the user. We implement a zero-shot baseline that scores an image by the cosine similarity between the image embedding and the text embedding of the desired concept. For evaluation metrics that require binary predictions, we classify an image as positive if the cosine similarity exceeds a certain threshold. When using CLIP, we chose the threshold to be 0.28, based on LAION-5B's human inspection [52]. We similarly chose 0.2 as a threshold when using ALIGN based on our own inspection. We also compare against a recently released active learning algorithm for learning rare vision categories [39]. This system is the most relevant related work, as described in Section 2. We replace our active learning algorithm with theirs and compare the performance in Section 4.4.

Evaluation protocol. To evaluate the models trained with the Agile Modeling prototype, we require an appropriate test set. Ideally, the user would provide a comprehensive test set—for example, ImageNet holds out a test set from their collected data [49]. However, since our users are volunteers with limited annotation time, they cannot feasibly label the entire LAION-400M dataset or its 100M test split. Additionally, since we are considering rare concepts, labeling a random subset of unlabeled images is unlikely to yield enough positives. To address these problems, we ran stratified sampling on each model, which divides images based on their model score into 10 strata ranging from $[0, 0.1)$ to $[0.9, 1.0]$. In each strata, we hash each image URL to a 64-bit integer using the pseudorandom function SipHash [3] and include the 20 images with the lowest hashes in the evaluation set. Each model contributes equally to final test set. The final evaluation set has over 500 images per category with approximately 50% positive rate. The full details of the evaluation set distribution and acknowledgement of its potential biases can be found in Appendix B.

Other hyperparameters. The text-to-image expansion expands each user-provided query to 100 nearest-neighbor images. Next, the image selection stage randomly selects a total of 100 images from all queries, leading to an initial training set of 100 samples for the first model. Users are asked to perform 5 rounds of active learning, rating 100 images per step. These hyperparameters were chosen based on two held-out concepts, and the ablation results in Section 4.4.

4.3. Results

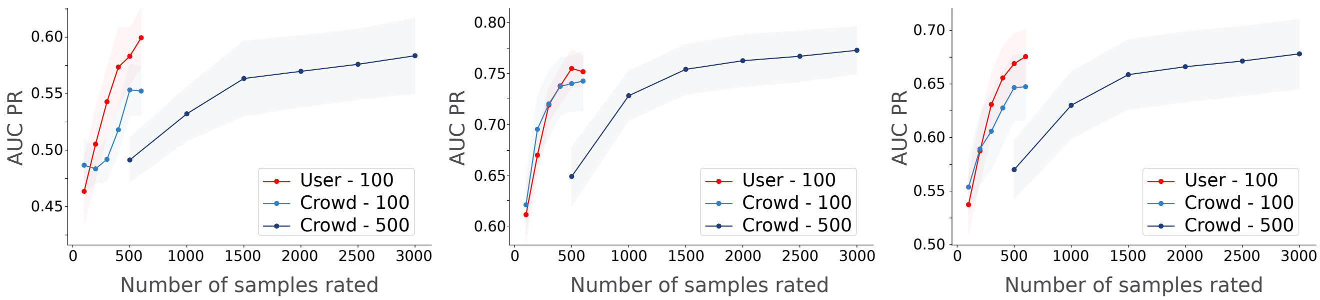
4.3.1 Users produce classifiers in minutes

A key value proposition of Agile Modeling is that the user should be able to train a model in minutes. We now report the feasibility of this proposition.

Measuring Time. The time it takes per for each step of the framework is detailed in Table 1. Our proposed Agile Modeling implementation trains one initial model and conducts five active learning rounds, taking 24 minutes on average to generate a final model.

Comparison with zero-shot. We start by comparing against zero-shot classification, which corresponds to a scenario with minimal effort from the user. In Figure 3, we present the performance our instantiations of the Agile Modeling framework against a zero-shot baseline across two image-text co-embeddings: CLIP [46] and ALIGN [24]. We find that the zero-shot performance is roughly on par as a supervised model trained on 100 labeled examples by the user. However, after the user spends a few more minutes rating (i.e., as the number of user ratings increases from 100 to 600), the resulting supervised model outperforms zero-shot.

User time versus performance. To measure the trade-off between user time versus model performance, we show in Figure 3 the AUC PR of the model across active learning rounds. We include additional metrics in Appendix C. We include results for both CLIP and ALIGN representations as input to our classifiers. We also compare against the respective zero-shot models using CLIP and ALIGN, which are considered the zero effort case. For both types of represen-



(a) Hard concepts – per # samples rated (b) Easy concepts – per # samples rated (c) All concepts – per # samples rated

Figure 4: Performance per # samples rated by the user or crowd. AUC PR mean and standard error over subsets of concepts: hardest for the zero-shot model (left), easiest for the zero-shot model (middle), all (right). Each ● represents an AL round. The three settings `User-100`, `Crowd-100`, `Crowd-500` refer to the rating setups described in Section 4.3.2.

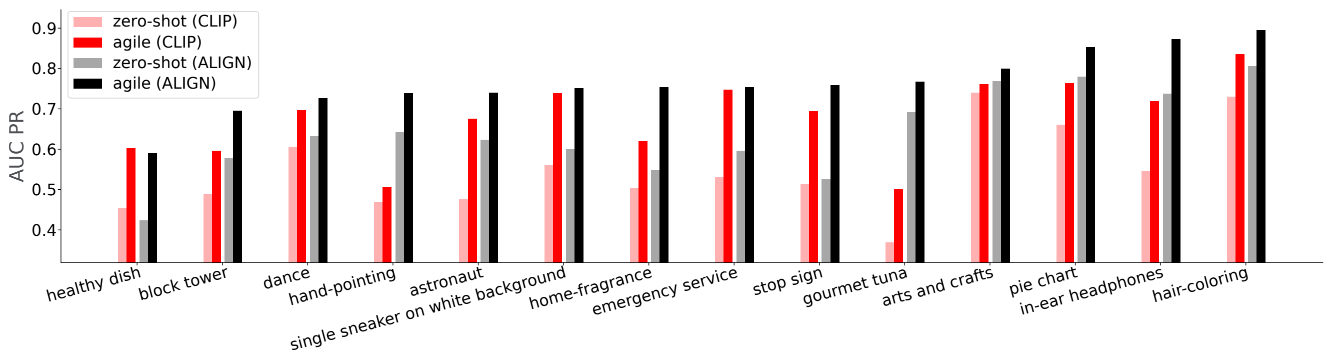


Figure 5: Model performance per concept for zero-shot and user-in-the-loop Agile models on CLIP and ALIGN embeddings.

tations, we see a steeper increase in performance for the first 3 active learning rounds, after which the performance starts to plateau, consistent with existing literature applying active learning to computer vision tasks [25]. Interestingly, for CLIP representations, the initial model trained on only 100 images performs worse than the zero-shot baseline, but the zero-shot model is outperformed with just one round of active learning. We do not see this effect on ALIGN representations, where even 100 samples are enough to outperform the zero-shot model—perhaps because ALIGN representations are more effective. We compare CLIP and ALIGN in more detail in Section 4.4. Importantly, we show that with only 5 minutes of the user's time (Table 1), we can obtain a model that outperforms the zero-shot baseline by at least 3%. After 24 minutes, this performance gain grows to 16%.

4.3.2 Value of users in the loop versus crowd workers

We now study the value of empowering users to train models by themselves. In particular, we address the following question: *Are there concepts for which a user-centered Agile framework leads to better performance?*

Users have an advantage over crowd raters in their ability to rate images according to their subjective specifications. However, this subjectivity, or "concept difficulty" varies by concept: if a concept is universally understood, the advantage diminishes. Conversely, complex, nuanced concepts are harder for crowd workers to accurately label. To take this into consideration, we first partition the concepts into two datasets based on their difficulty, using zero-shot performance as a proxy for concept difficulty. The 7 concepts that admit the highest zero-shot performance are considered "easy," while the remaining 7 concepts are considered "hard." The specific groups can be found in Appendix D. Notice that the "difficult" concepts include more subjective concepts such as `gourmet tuna` (illustrated in Figure 1), or concepts with multiple and ambiguous visual modes such as `healthy dish`; whereas the "easy" concepts include simple, self-explanatory concepts such as `dance` or `single sneaker on white background`.

We then evaluate models trained by three sets of raters:

1. `User-100`: Users rate 100 images for the initial model and every AL round (total 600 images).
2. `Crowd-100`: Crowd workers rate 100 images for the initial model and every AL round (total 600 images).
3. `Crowd-500`: Crowd workers rate 500 images for the initial model and every AL round (total 3000 images).

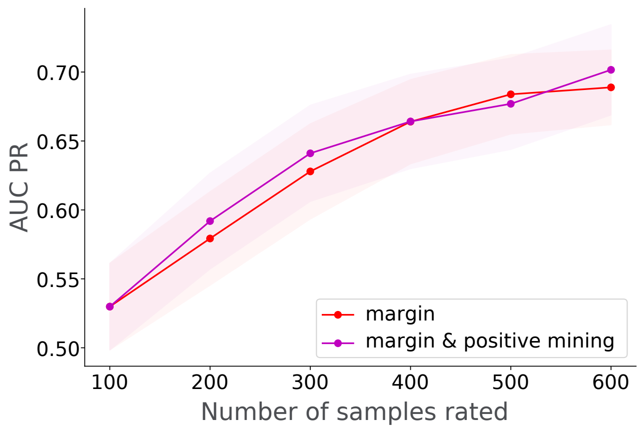


Figure 6: Model performance for two active learning methods: margin and the approach of [39] (margin & positive mining). Each • corresponds to an AL round. We show the AUC PR mean and standard error over all concepts.

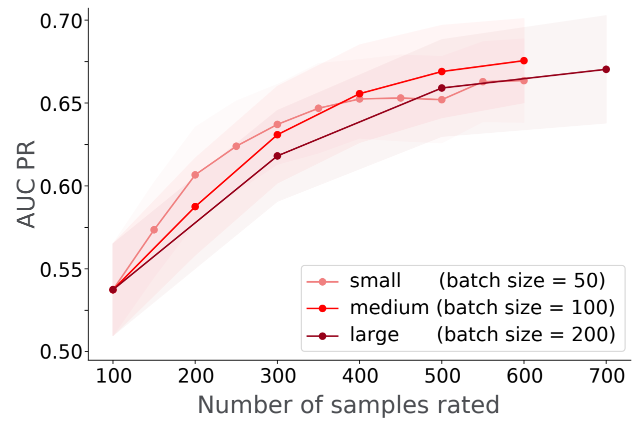


Figure 7: Model performance during active learning with 3 AL batch sizes: small (50), medium (100), large (200). Each • corresponds to an AL round. We show the AUC PR mean and standard error over all concepts.

The only difference in the configurations above is who the raters are (user or crowd) and the total number of ratings. For crowd ratings, having clear instructions is crucial for accurate results, but obtaining them is a non-trivial task in the machine learning process [19, 14]. In this experiment, crowd workers read instructions created by the users, who noted difficult cases that they found during labeling. Details about the crowd instructions can be found in Appendix E.

We plot the results in Figure 4, which shows the average performance for the "hard", "easy" and all concepts as a function of the number of rated samples, using CLIP embeddings. Per-concept results can be found in Appendix F. On hard concepts, models trained with users (`User-100`) outperform models trained with crowd raters, even when $5\times$ more ratings are obtained from the crowd (`Crowd-500`). This suggests that Agile Modeling is particularly useful for harder, more nuanced and subjective concepts.

4.4. Ablation studies

Although our main contribution is introducing the problem of Agile Modeling, instantiating our prototype explores a number of design decisions. In this section, we lay out how these designs change the outcome.

Active learning method. Throughout the paper, we instantiate the active learning component with the well-known margin sampling method [50]. We now compare it to the active learning method used in Mullapudi et al [39]. We ran a version of our instantiation of the Agile framework where we replace margin with the combined margin and positive mining strategy chosen by [39] and described in Section 3.2. The performance of the two methods per AL round is shown in Figure 6. Interestingly, despite the fact that Mullapudi et al. [39] introduced this hybrid approach to improve upon margin sampling, in this setting the two methods perform similarly across all AL rounds. We see the same effect on most concepts when inspecting on a per-concept basis in Appendix C.2. One potential explanation for this is that the initial model trained before AL is already good enough (perhaps due to the powerful CLIP embeddings) for margin sampling to produce a dataset balanced in terms of positive and negative, and thus explicitly mining easy positives as in [39] is not particularly useful. Since the two methods perform equivalently, we opted for the simpler and more efficient one (i.e., margin) in the rest of the experiments.

Active learning batch size. Our prototype asks the user to annotate images across 5 rounds of active learning, 100 images per round. However, we can simultaneously change the number of images rated per round and the number of active learning rounds the user conducts. We evaluate the downstream effects of changing active learning batch size and number of rounds on model performance and time spent. We consider 3 batch sizes: small (50 images/batch), medium (100 images/batch), large (200 images/batch). We run repeated rounds of active learning with each of these settings, retraining the model after each round using CLIP representations. The results in Figure 7 show that, for a fixed amount of images rated, smaller batch sizes are better than larger, especially so in the beginning. This result is expected, because for a fixed rating budget, the smaller batch setting has the chance to update the model more frequently. While these results suggest that we should opt for a smaller batch size, there is still a trade-off between user time and performance, even when we have the same total number of samples rated. That is because model training takes about 1-2 minutes during which the user is idle, and so smaller batch sizes lead to longer time investment from the users.

As a good compromise, we chose 100 as our batch size.

Stronger pretrained model improves performance. Since our system leverages image-text co-embeddings to find relevant images and quickly train classifiers, a logical question is: how does changing the underlying embedding change the performance of the classifier? To do this, we compare CLIP versus ALIGN as the underlying embedding by replacing our pre-cached CLIP embeddings with ALIGN. We find that, with ALIGN, the AUROC of the final Agile model increased from 0.72 to 0.80 with a relative gain of 11.5%. The AUPR increased from 0.68 to 0.76, a relative gain of 13.1%. Furthermore, as Figure 5 demonstrates, both the ALIGN zero-shot and Agile models outperform their CLIP counterparts for almost every concept. This shows that building stronger image-text co-embeddings is foundational to improving the Agile Modeling process.

5. Experiments with ImageNet21k

Our user study validates the Agile Modeling framework on a small number of concepts over a web-scale unlabeled dataset. Now, we confirm that our framework can be effectively applied across a larger number of concepts to achieve significant improvements over zero-shot baselines. Due to the scale of this experiment, we simulate the user annotations using a fully-labeled dataset.

Experimental setup. We use the ImageNet21k dataset [13] which contains 21k classes and over 14M images. Out of these we select a subset of both easy and difficult classes, as described below. Each class corresponds to a binary classification problem as before. We apply the Agile Modeling framework with the ImageNet21k training set as the unlabeled data pool, and the test set for evaluation. Ground-truth class labels included in the dataset simulate a user providing ratings. Since the Agile Modeling process starts at concept definition with no labeled data, we use the class name and its corresponding WordNet [37] description as positive text phrases in the text-to-image expansion step. As before, we use a batch size of 100 and 5 rounds of active learning. We use ALIGN embeddings.

Concept selection. We use a subset of 100 of the 21k concepts for evaluation. 50 "easy" concepts are selected at random from the ImageNet 1000 class list. Additionally, we aim to replicate the ambiguity and difficulty of our original concepts by carefully selecting 50 further concepts with the following criteria based the WordNet lexicographical hierarchy: (1) 2-20 hyponyms, to ensure visual variety, (2) more than 1 lemma, to ensure ambiguity, (3) not an animal or plant, which have objective descriptions. Of the 546 remaining concepts, our 50 "hard" concepts are selected at random. The full list of chosen concepts is in Appendix H.

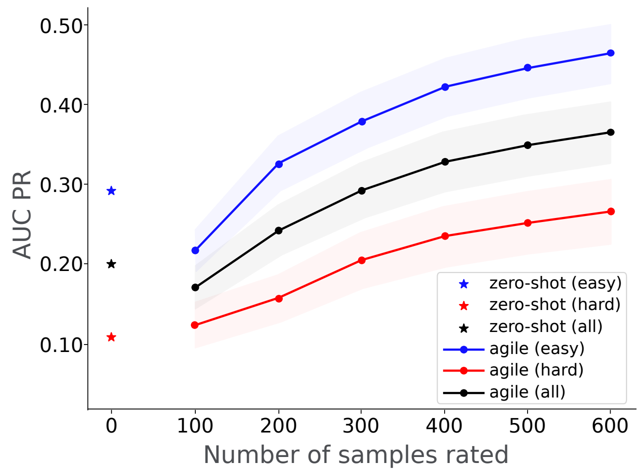


Figure 8: Model performance per amount of samples on ImageNet21k for both easy and hard classes (AUC PR mean and std error over classes). Each • represents an AL round.

Results. In Figure 8 we show the results of applying the Agile Modeling framework to ImageNet21k. We see a similar trend to our user experiments, with significant improvements over zero-shot baselines as well as continued improvement with each active learning round. We further observe that the "easy" concepts attained higher scores after the Agile Modeling process than the "hard" concepts. The zero-shot baseline differed significantly between the "easy" and "hard" concepts with scores of 0.29 and 0.11, respectively. The equivalent of 30 minutes of human work yields a 20% boost in AUC PR over the zero-shot baseline.

6. Discussion & conclusion

In this work, we promote the idea of empowering users without machine learning or engineering experience to create their own image classifiers for any concepts they might have need of in their daily life. We formalized this as the *Agile Modeling* problem—the process of turning any visual concept from an idea into a trained image classifier. We showed that, by using the latest advances in image-text pretrained models, we were able to initialize, train, and perform active learning in just a few minutes, enabling real-time user interaction for rapid model creation in less than 30 minutes. Through a set of experiments with 14 users, each modeling their own concept, we showed that our solution was able to quickly learn from users to create high performing classifiers even on difficult concepts. We also demonstrated the value of involving the users directly in the modeling process, by showing that models trained with the user-in-the-loop outperform those trained with crowd annotations, especially for very subjective concepts. We hope that our work showcases the opportunities and challenges of Agile Modeling and encourages future efforts.

References

[1] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120, 2014. 2

[2] Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds. In *International Conference on Learning Representations*, 2019. 3

[3] Jean-Philippe Aumasson and Daniel J Bernstein. SipHash: a fast short-input PRF. In *International Conference on Cryptology in India*, pages 489–508. Springer, 2012. 6

[4] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in Neural Information Processing Systems*, 33:22243–22255, 2020. 3

[5] Galen Chuang, Giulia DeSalvo, Lazaros Karydas, Jean-Francois Kagy, Afshin Rostamizadeh, and A Theeraphol. Active learning empirical study. In *NeurIPS 2019 Workshop on Learning with Rich Experience: Integration of Learning Paradigms*, 2019. 3

[6] Gui Citovsky, Giulia DeSalvo, Claudio Gentile, Lazaros Karydas, Anand Rajagopalan, Afshin Rostamizadeh, and Sanjiv Kumar. Batch Active Learning at Scale. In *Advances in Neural Information Processing Systems*, 2021. 3

[7] Niv Cohen, Rinon Gal, Eli A Meirom, Gal Chechik, and Yuval Atzmon. "This is my unicorn, Fluffy": Personalizing frozen vision-language representations. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XX*, pages 558–577. Springer, 2022. 2

[8] Cody Coleman, Edward Chou, Julian Katz-Samuels, Sean Culatana, Peter Bailis, Alexander C Berg, Robert Nowak, Roshan Sumbaly, Matei Zaharia, and I Zeki Yalniz. Similarity search for efficient active learning and search of rare concepts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6402–6410, 2022. 3

[9] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via Proxy: Efficient Data Selection for Deep Learning. *arXiv preprint arXiv:1906.11829*, 2019. 3

[10] Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *AAAI*, volume 5, pages 746–751, 2005. 4

[11] Maureen Daum, Enhao Zhang, Dong He, Magdalena Balazinska, Brandon Haynes, Ranjay Krishna, Apryle Craig, and Aaron Wirsing. VOCAL: Video Organization and Interactive Compositional AnaLytics. In *12th Annual Conference on Innovative Data Systems Research (CIDR'22)*, 2022. 2

[12] Maureen Daum, Enhao Zhang, Dong He, Brandon Haynes, Ranjay Krishna, and Magdalena Balazinska. VOCALExplore: Pay-as-You-Go Video Data Exploration and Model Building. *arXiv preprint arXiv:2301.00929*, 2023. 3

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2009. 1, 9

[14] Steven Dow, Anand Kulkarni, Scott Klemmer, and Björn Hartmann. Shepherding the crowd yields better work. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1013–1022, 2012. 8

[15] Jerry Alan Fails and Dan R Olsen Jr. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent User Interfaces*, pages 39–45, 2003. 2

[16] Li Fei-Fei and Ranjay Krishna. Searching for Computer Vision North Stars. *Daedalus*, 151(2):85–99, 2022. 4

[17] Rebecca Fiebrink, Perry R Cook, and Dan Trueman. Play-along mapping of musical controllers. In *ICMC*, 2009. 2

[18] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017. 4

[19] Ujwal Gadiraju, Jie Yang, and Alessandro Bozzon. Clarity is a worthwhile quality: On the role of task clarity in microtask crowdsourcing. In *Proceedings of the 28th ACM Conference on Hypertext and Social Media*, pages 5–14, 2017. 8

[20] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An Image is Worth One Word: Personalizing Text-to-Image Generation using Textual Inversion. *arXiv preprint arXiv:2208.01618*, 2022. 2

[21] Mitchell L Gordon, Kaitlyn Zhou, Kayur Patel, Tatsunori Hashimoto, and Michael S Bernstein. The disagreement deconvolution: Bringing machine learning performance metrics in line with reality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2021. 1, 5

[22] Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. Quantization based fast inner product search. In *Artificial Intelligence and Statistics*, pages 482–490. PMLR, 2016. 4

[23] Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer, 2001. 4

[24] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916. PMLR, 2021. 2, 3, 4, 6

[25] Siddharth Karamcheti, Ranjay Krishna, Li Fei-Fei, and Christopher D Manning. Mind Your Outliers! Investigating the Negative Impact of Outliers on Active Learning for Visual Question Answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7265–7281, 2021. 7

[26] Mina Khan, P Srivatsa, Advait Rane, Shriram Chenniappa, Asadali Hazariwala, and Pattie Maes. Personalizing pre-trained models. *arXiv preprint arXiv:2106.01499*, 2021. 2, 3

[27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 6

[28] W Bradley Knox and Peter Stone. Learning non-myopically from human-generated reward. In *Proceedings of the 2013 international conference on Intelligent User Interfaces*, pages 191–202, 2013. 2

[29] Ivan Krasin, Tom Duerig, Neil Alldrin, Andreas Veit, Sami Abu-El-Haija, Serge Belongie, David Cai, Zheyun Feng, Vittorio Ferrari, Victor Gomes, Abhinav Gupta, Dhyanesh Narayanan, Chen Sun, Gal Chechik, and Kevin Murphy. OpenImages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://github.com/openimages*, 2016. 1

[30] Ranjay Krishna, Mitchell Gordon, Li Fei-Fei, and Michael Bernstein. Visual intelligence through human interaction. *Artificial Intelligence for Human Computer Interaction: A Modern Approach*, pages 257–314, 2021. 2

[31] Ranjay Krishna, Donsuk Lee, Li Fei-Fei, and Michael S Bernstein. Socially situated artificial intelligence enables learning from human interaction. *Proceedings of the National Academy of Sciences*, 119(39):e2115730119, 2022. 2

[32] Matthew Lease. On quality control and machine learning in crowdsourcing. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*. Citeseer, 2011. 1

[33] David Lewis and William Gale. A sequential algorithm for training text classifiers. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994. 3

[34] David D Lewis. A sequential algorithm for training text classifiers: Corrigendum and additional data. In *ACM SIGIR Forum*, volume 29, pages 13–19. ACM New York, NY, USA, 1995. 4

[35] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014. 1

[36] Robert Loftin, Bei Peng, James MacGlashan, Michael L Littman, Matthew E Taylor, Jeff Huang, and David L Roberts. Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning. *Autonomous Agents and Multi-Agent Systems*, 30:30–59, 2016. 2

[37] George A Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995. 9

[38] Ravi Teja Mullapudi, Fait Poms, William R Mark, Deva Ramanan, and Kayvon Fatahalian. Background splitting: Finding rare classes in a sea of background. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8043–8052, 2021. 3

[39] Ravi Teja Mullapudi, Fait Poms, William R Mark, Deva Ramanan, and Kayvon Fatahalian. Learning Rare Category Classifiers on a Tight Labeling Budget. In *IEEE/CVF International Conference on Computer Vision*, pages 8423–8432, 2021. 2, 5, 6, 8, 15, 16

[40] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 2022. 2

[41] Junwon Park, Ranjay Krishna, Pranav Khadpe, Li Fei-Fei, and Michael Bernstein. AI-based request augmentation to increase crowdsourcing participation. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 7, pages 115–124, 2019. 2

[42] Kayur Patel, Naomi Bancroft, Steven M Drucker, James Fogarty, Amy J Ko, and James Landay. Gestalt: Integrated Support for Implementation and Analysis in Machine Learning. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 37–46, 2010. 2

[43] Genevieve Patterson, Grant Van Horn, Serge Belongie, Pietro Perona, and James Hays. Tropel: Crowdsourcing Detectors with Minimal Training. In *Third AAAI Conference on Human Computation and Crowdsourcing*, 2015. 2

[44] Robert Pinsler, Jonathan Gordon, Eric Nalisnick, and José Miguel Hernández-Lobato. Bayesian Batch Active Learning as Sparse Subset Approximation. *Advances in Neural Information Processing Systems*, 32, 2019. 3

[45] Sarah Pratt, Rosanne Liu, and Ali Farhadi. What does a platypus look like? Generating customized prompts for zero-shot image classification. *arXiv preprint arXiv:2209.03320*, 2022. 3

[46] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 2, 3, 4, 6

[47] A Ratner, S.H Bach, H Ehrenberg, J Fries, S Wu, and C Re. Snorkel: Rapid Training Data Creation with Weak Supervision. In *VLDB Endowment*, pages 269–282, 2017. 2

[48] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A Survey of Deep Active Learning. *ACM Computing Surveys (CSUR)*, 54(9):1–40, 2021. 4

[49] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 6

[50] Decomain C Scheffer, T and S Wrobel. Active Hidden Markov Models for Information Extraction. In *International Conference on Advances in Intelligent Data Analysis (CAIDA)*, page 309–318, 2001. 8

[51] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active Hidden Markov Models for Information Extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer, 2001. 4

[52] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo

Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. LAION-5B: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022. 6, 15

[53] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. LAION-400M: Open Dataset of CLIP-Filtered 400 Million Image-Text Pairs. *arXiv preprint arXiv:2111.02114*, 2021. 5

[54] Ozan Sener and Silvio Savarese. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *International Conference on Learning Representations*, 2018. 3

[55] Burr Settles. Active Learning Literature Survey. *Computer Sciences Technical Report 1648, University of Wisconsin, Madison*, 2010. 3

[56] Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. Deep Active Learning for Named Entity Recognition. *arXiv preprint arXiv:1707.05928*, 2017. 3

[57] Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 614–622, 2008. 1

[58] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in Neural Information Processing Systems*, 30, 2017. 4

[59] Otilia Stretcu, Edward Vendrow, Kenji Hata, Krishnamurthy Viswanathan, Vittorio Ferrari, Sasan Tavakkol, Wenlei Zhou, Aditya Avinash, Enming Luo, Neil Gordon Alldrin, MohammadHossein Bateni, Gabriel Berger, Andrew Bunner, Chun-Ta Lu, Javier A Rey, Giulia DeSalvo, Ranjay Krishna, and Ariel Fuxman. A prototype implementation for Agile Modeling. https://github.com/google-research/google-research/tree/master/agile_modeling, 2023. 2, 5

[60] Andrea L Thomaz and Cynthia Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172(6-7):716–737, 2008. 2

[61] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. YFCC100M: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016. 2

[62] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *European Conference on Computer Vision*, pages 266–282. Springer, 2020. 3

[63] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples. *arXiv preprint arXiv:1903.03096*, 2019. 3

[64] Paroma Varma, Bryan D He, Payal Bajaj, Nishith Khandwala, Imon Banerjee, Daniel Rubin, and Christopher Ré. Inferring generative model structure with static analysis. *Advances in Neural Information Processing Systems*, 30, 2017. 2

[65] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching Networks for One Shot Learning. *Advances in Neural Information Processing Systems*, 29, 2016. 3

[66] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020. 4

[67] Xiang Wu, Ruiqi Guo, Ananda Theertha Suresh, Sanjiv Kumar, Daniel N Holtmann-Rice, David Simcha, and Felix Yu. Multiscale quantization for fast similarity search. *Advances in Neural Information Processing Systems*, 30, 2017. 4

[68] Enhao Zhang, Maureen Daum, Dong He, Brandon Haynes, Ranjay Krishna, and Magdalena Balazinska. EQUI-VOCAL: Synthesizing Queries for Compositional Video Events from Limited User Interactions. *arXiv preprint arXiv:2301.00929*, 2023. 2