Crowdsourcing and Its Applications in Computer Vision

Catherine Wah
University of California, San Diego
cwah@cs.ucsd.edu

May 26, 2011

Abstract

Crowdsourcing has emerged in the past decade as a popular model for online distributed problem solving and production. The creation of Amazon Mechanical Turk (MTurk) as a "micro-task" marketplace has facilitated this growth by connecting willing workers with available tasks. Within computer vision, MTurk has proven especially useful in large-scale image label collection, as many computer vision algorithms require substantial amounts of training data. In this survey, we discuss different types of worker incentives, various considerations for MTurk task design, methods for annotation quality analysis, and cost-effective ways of obtaining labels in a selective manner. We present several examples of how MTurk is being utilized in the computer vision community. Finally, we discuss the implications that MTurk usage will have on future computer vision research.

1 Introduction

The term *crowdsourcing*, coined in 2006 [14], refers to the business model in which jobs are outsourced to "an undefined, generally large group of people in the form of an open call" [15]. One example of crowdsourcing is distributed large-scale human-based computation, which involves outsourcing various steps in a computational process to humans [47].

In this survey, we examine in depth Amazon Mechanical Turk [1], a popular crowdsourcing platform that is intended for tasks that benefit from human intelligence. Using an API, *requesters* post tasks called *human intelligence tasks* (HITs) to MTurk. *Workers*, also known as "Turkers," can then find available HITs by searching the marketplace and selecting the ones they want to complete for a nominal payment, typically on the order of cents. If multiple responses to a certain HIT are needed, for example for majority voting or consensus, one can specify the number of *assignments* per HIT, or the total number of unique workers who can complete a specific HIT. At a recent count, these MTurk workers numbered over $200,000$, representing 185 countries [12].

MTurk has become very popular within the computer vision community [34] as a venue for completing image labeling tasks; it can be a practical solution for obtaining large quantities of annotations, since many different types of computer vision algorithms, such as object detection, tracking, and recognition, require substantial amounts of labeled training data. As a result, several predominant areas of research in crowdsourcing that pertain to computer vision have emerged. Our goal in this work is to examine crowdsourcing strategies for computer vision applications as they would relate to a researcher/requester. From a requester point of view, there are a number of questions to consider in MTurk task creation:

- **What motivates workers to complete tasks?** One needs to attract a willing and available labor pool in order to get tasks completed in a timely manner.
- **What is the best way to design a task?** Given a task, it is necessary to consider what design patterns or workflow will enable successful and efficient completion of the task by Turkers.

[1] http://www.mturk.com/

- **What is the best way to ensure high quality results?** As a requester, one wishes to ensure that results are of high quality. Managing quality for simple tasks is relatively straightforward, but for complex tasks, we need to account for variables such as task difficulty, worker expertise, and worker bias.

- **How can one use crowdsourcing in a cost effective manner?** While crowdsourcing is relatively inexpensive, it is not free. We wish to minimize crowdsourcing costs while maintaining sufficiently high quality in results. For example, certain types of image annotations are more useful than others in training a classifier. At the same time, certain images may be more difficult for humans to label with particular annotations (e.g. providing a segmentation for images with cluttered backgrounds).

These concerns and the domains that encompass them are all intertwined with one another: worker incentives can play a role in task design, which influences the quality of results and deployment costs, while quality and cost are also inherently correlated.

In this survey, we focus on how the payment-based model of MTurk has affected the landscape of crowdsourcing in the context of computer vision applications, and we examine different strategies for addressing the aforementioned questions. This is not intended to be a comprehensive summary of current research directions in crowdsourcing; we highlight some recent advances in the field and discuss implications for future computer vision research. For instance, we do not cover MTurk marketplace demographics, human computation game design, cognitive aspects of crowdsourcing, or crowdsourcing marketplaces other than MTurk.

We begin by discussing in Section 2 factors that motivate workers to perform tasks. In Section 3, we study the role experimental design has in MTurk task completion. In Section 4, we examine different models for the human annotation process. We consider various methods for obtaining labels selectively and cost efficiently in Section 5. In Section 6, we present examples of crowdsourcing as it has been applied to tasks in computer vision, and finally, in Section 7, we discuss and evaluate the utility and effectiveness of crowdsourcing.

2 Task Incentives

The incentives that motivate workers to complete tasks play a significant role in the successful implementation of crowdsourcing, as it is necessary to attract an appropriate and willing labor pool. We briefly present several methods for incentivizing tasks and corresponding examples found in computer vision.

2.1 Entertainment

One approach incentivizes labeling tasks by presenting them as games; users are motivated by personal enjoyment or social reward. For example, some of the Games With A Purpose (GWAPs) [37] encapsulate various modalities of image annotation (free text, object category label, segmentation, etc.) [37, 40, 38, 39]. In these cases, humans perform labeling tasks for free.

2.2 Altruism

Another incentive arises from altruism, in which users are motivated by the idea that they can contribute to the "greater good." One example is LabelMe [30], an image database populated with annotations collected with a web-based tool; labels are provided on a volunteer basis.

Altruistic incentives are particularly evident in the scientific community in the form of *citizen science*, in which people are collaborators who are "proactively involved in the process of science by participating," versus just being "a member of the crowd" [48]. A notable goal of citizen science is to promote public understanding of science, and thus there usually is an educational component to these citizen science endeavors. Examples include GalaxyZoo [2] and the Valley of the Khans [3] project, in which volunteers perform image labeling tasks.

Another means of engaging workers is to appeal to their self-interested curiosity, in addition to any altruistic motivations. For example, one could provide a service in the form of an interactive field guide [3, 28]. Users who wish to identify unknown objects (e.g. bird species, Manhattan buildings, plant species, etc.) can submit a query by uploading an image. The computer attempts to identify the object by using machine vision algorithms and "hints"

[2] http://www.galaxyzoo.org
[3] http://exploration.nationalgeographic.com/mongolia/home

provided by humans in the form of responses to questions about the object.

When users engage in this interactive process, they assist in improving the system by contributing images and corresponding labels, while additionally gaining new knowledge when objects are correctly categorized; in this way, the users serve as citizen scientists. Therefore, this interactive approach is mutually beneficial to both humans and machine vision algorithms. We further discuss an example of an interactive field guide in Section 6.2.

2.3 Financial Reward

An alternate solution to these volunteer efforts is to replace the internal motivation of humans with monetary payment, as with the MTurk paradigm; numerous examples are studied in Section 6. Of particular interest is the relationship between the financial incentives of crowdsourcing and worker performance: does increasing the rate of compensation for a given task lead to better results? Mason and Watts [23] discovered that higher pay increases work quantity but has no significant effect on quality. The increased pay rate makes the HIT more appealing to Turkers and thus they complete more HITs.

Mason and Watts [23] also investigated the payment perceptions of MTurk workers and observed that Turkers consistently valued their own work above the current pay rate. Furthermore, in comparing performance with payment versus no payment at all, they observed little impact on accuracy; workers perceive the value of work as being correlated with the actual pay rate.

2.4 Discussion

Incentives can influence elements of task design, including choices in user interface and visual appearance. Generally, tasks are designed with multiple incentives to attract workers. Certain citizen science projects are formulated as involved, immersive games, as it is necessary to maintain worker interest while training them to perform more complex tasks such as protein folding [5]. In training workers to perform tasks, one also enables users to gain new skills or knowledge.

In addition, we note that financial incentives have little effect on work quality, and increasing monetary rewards is only effective when one requires data to be labeled quickly. This is further demonstrated by Horton and Chilton [13] in their estimate of MTurk reservation wages. In labor economics this term refers to the minimum wage rate at which a worker would be willing to accept a job, controlling for work type and work conditions; for MTurk workers, the median reservation wage is calculated to be $1.38/hour. In comparison, Ipeirotis [19] has estimated the median hourly wage for MTurk tasks to be $4.80/hour, based on observations of HIT arrival and completion rates.

3 Experimental Design

In order to obtain the best results from workers, it is important to determine the optimal design of the task in question. The MTurk platform allows requesters to specify HIT settings, worker qualifications, and HIT layout design [4]. We discuss in Section 3.1 the different dimensions of HIT design, and in Section 3.2 we explore in more depth various HIT workflow strategies.

3.1 Task Parameter Selection

One dimension of experimental design encompasses the specification of HIT parameters, without modifying the HIT layout. For any given HIT, a requester can specify: the maximum time a worker is allotted to complete the HIT; the length of time the HIT is available on the marketplace; the amount of reward; and the number of assignments per HIT. A requester can also set required qualifications for workers, in order to restrict the HIT availability. These can include geographic constraints, a minimum approval rating (percentage of completed HITs in which the worker's responses were approved by requesters), or the successful completion of qualification tests that can be used to "train" workers to perform certain tasks [4].

In addition to specifying the amount of reward per HIT, a requester can use basic compensation strategies, encouraging good results by giving bonuses or penalizing bad work by rejecting the worker's submission. This requires attentive moderation of worker quality, done manually or automatically (see Section 4).

Designing the task interface requires taking into account the ergonomics of the HIT; the visual appearance

should enable Turkers to quickly understand and complete the requested task. For a simple image labeling task, one can present a gallery of images that allows workers to select and label multiple images at once; for more complex labeling tasks, images can be displayed one at a time. Additionally, it is advantageous to adopt a defensive approach to task interface design, such that accurate task completion requires as much effort as adversarial responses [21]. Commonly, these ergonomics are determined empirically and are tailored for the task at hand.

Huang et al. [16] advocate a design approach that involves learning a task-specific model of the effect of design on the work output quality. These models are then used to generate good task designs, which can be evaluated based on: (1) the amount of work completed within a certain time frame, and (2) the trade off between work completed per dollar spent and the rate of work. By learning what parameters are optimal for a particular task, we can effectively distribute the work across subtasks that can be performed in parallel (e.g. divide up a multi-image labeling HIT into multiple single-image HITs).

3.2 Human Computation Process Design

A more complex dimension of HIT design is the paradigm in which humans are added to a human computation process, enabling different problems to be solved in novel ways. Human computation processes are still not very well understood; by breaking down these processes into discrete components, we can potentially make the processes more efficient and apply them to a wider range of problems. We discuss two notable design patterns that have been proposed for executing tasks on MTurk.

3.2.1 Iterative versus Parallel Processes

Little et al. [22] describe how human computation processes can be broken down into two types: creation and decision tasks. *Creation tasks* solicit the user for new content, in the form of descriptions, ideas, solutions, etc. The goal is to generate new content of high quality, and usually there are fewer constraints on user input. On the other hand, *decision tasks* solicit the user for an opinion about existing content (for example, comparison or rating tasks), where the objective is to obtain an accurate response. Decision tasks may ask for multiple responses and use the aggregate or average.

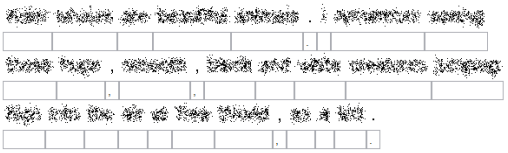

Figure 1: Little et al. [22] observed the effect of parallel and iterative human computation processes on response quality. In a transcription task, Turkers were able to recognize all but one of the words in this dithered sentence: "Killer whales are beautiful animals. I remember seeing these huge, smooth, black and white creatures jumping high into the air at Sea World, as a kid" (source: [22]).

Creation and decision tasks can be combined into a sequence of tasks to form an iterative or parallel process. An *iterative process* consists of a sequence of creation tasks; if it is not easy to merge the results of multiple creation tasks, one can add a comparison task in between creation tasks in order to keep track of the best result so far. This process allows workers to vote on and iteratively improve on work done by other workers. A *parallel process* consists of a series of creation tasks, performed in parallel; the best result is found using a sequence of comparison tasks.

Little et al. compared these processes for different problem domains, including transcription (of dithered text; see Figure 1) and writing (image descriptions). Transcription problems have determinate answers, whereas writing tasks are open-ended and must be judged subjectively. For the iterative implementations of these tasks, workers were able to see previous workers' proposed transcriptions or descriptions. In order to determine quality, results were compared to the ground-truth text in the transcription task and were rated by other Turkers in the description task.

In applying iterative processes, Little et al. [22] observed increased average quality in results over parallel processes but also lower variance. This suggests that showing Turkers previous work in some cases can negatively affect quality by leading them down the wrong path.

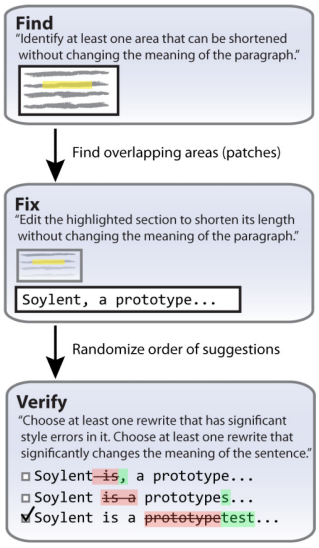

Figure 2: Bernstein et al. [1] proposed the Find-Fix-Verify design pattern for deploying tasks on MTurk. An example of a proofreading task is shown (source: [1]).

3.2.2 Find-Fix-Verify Pattern

Building upon this work, Bernstein et al. [1] proposed the Find-Fix-Verify pattern as a design paradigm intended for open-ended tasks. Among Turkers there is a high variance in the amount of effort they invest in a given task, and additionally, they may introduce errors when working on a complex task; Bernstein et al. estimate that 30% of results for open-ended tasks are of poor quality. Find-Fix-Verify (Figure 2) enables workers to make clear contributions regardless of their reliability.

The first stage of this pattern, Find, asks Turkers to find portions of the original work that require attention; for example, in a proofreading task, Turkers are asked to identify passages that need editing. Aggregation of these multiple independent results then allows us to identify consistent problems. The consistently flagged patches are fed in parallel to the Fix stage.

The Fix stage has Turkers revise the identified portions. Each task is restricted to a certain area of interest; for instance, the worker can see the entire paragraph in a proofreading task but is only allowed to edit the text in the identified patch. For each patch, a small number of Turkers propose revisions; this is sufficient to generate viable suggestions.

Finally, during the Verify stage, additional workers vote on the suggested revisions from the previous stage. Depending on the task, they can vote on the best option or flag poor suggestions. Turkers from the Fix stage are excluded from working on tasks from this stage, to prevent them from voting on their own work.

The advantage of dividing up a task in this manner allows one to take advantage of different labeler reliabilities. For example, certain "lazy" Turkers may be not be thorough in their work, and in the isolated Fix stage one can request they make revisions to specific portions that they may have otherwise ignored, had the Find and Fix stages been combined. This separation of the Find and Fix stages enables parallel merging of edits, and the Verify stage reduces noise in responses. In this way one has more precise control over the amount of worker contribution.

3.3 Discussion

These patterns are fairly basic and as building blocks for HIT design are able to represent a range of different tasks. The implementation of these building blocks is an open problem; for example, a decision task can be achieved using relative rankings, pair-wise comparisons, or absolute rankings. Moreover, for more complex tasks, other building blocks and patterns may be needed.

The Find-Fix-Verify pattern is a promising generalizable approach for completing a variety of open-ended tasks. It does not train previous workers but rather recruits workers on demand to perform personalized tasks and thus has the capability of operating in real-time. However, the process can be slowed down due to lag time in workers accepting and completing the posted tasks.

In addition, cost becomes a more conspicuous factor in processes such as Find-Fix-Verify that select one best solution from multiple responses. Much work gets discarded in these voting or comparison tasks, and it remains to be seen whether productivity gains from these design processes are worth the trade off in expense.

One disadvantage of breaking tasks down into smaller blocks is that they are removed from their original context [10]. This did not appear to influence results for word pro-

cessing tasks [1], which usually can be isolated to phrases or sentences, but may be of more relevance for visually oriented tasks that require making decisions or modifications at a holistic level, e.g. visual page layout.

Overall as Little et al. [22] noted, a collective approach to problem solving that makes use of different HIT building blocks can be very beneficial, as good solutions are shared among workers. In this way, iterative processes appear to offer some performance advantages over parallel processes. However, workers can also prematurely converge on suboptimal solutions, resulting in less variability in responses, which may be undesirable for tasks that involve collecting as many unique, high quality labels as possible, such as image tagging.

4 Quality Management

In order to make use of results obtained through MTurk, one must be able to guarantee a certain degree of accuracy, especially when there is noise in user responses and the ground truth is unknown. This involves identifying good annotations and throwing out bad ones, a task that quickly becomes challenging to manage when working with hundreds of thousands of labels. In this section, we discuss various forms of quality management for MTurk tasks. There are several heuristic strategies for reviewing MTurk results that are discussed in Section 4.1. We also review more sophisticated approaches that aim to model the annotation process in a Bayesian framework.

4.1 Heuristics

Using heuristics for MTurk quality management is a simple way to significantly improve results. These include tracking Turker answer histories to identify trusted individuals who consistently provide high-quality responses. These expert workers can then be asked to complete tasks or review the results of other workers [4].

When experts are not identified in the labor pool, other approaches are applicable, such as forced agreement (e.g. the ESP Game [37]); using control questions in which the answers are already known and are treated as the "gold standard" [32]; and checking responses from multiple Turkers against one another. One example of the latter is averaging multiple naïve labels provided by non-experts, a technique that yields quality comparable to a few expert annotations [33].

Using this type of majority vote heuristic is relatively easy to implement but requires a large number of labels, which are individually cheap but can accumulate in cost very quickly. Sheng et al. [31] investigated when it was worthwhile to request repeated labels for training examples and demonstrated improved quality in results when selectively acquiring multiple labels.

4.2 Modeling the Human Annotation Process

For more complex tasks, quality management becomes more involved and we require more effective methods. Much work has focused on developing models [33, 16, 20, 24] that represent certain aspects of the image formation and human annotation process, including: worker expertise, worker bias, worker strengths, label value, and task difficulty.

Dawid and Skene [6] used full confusion matrices for each labeler to account for labeler ability and bias. They estimated the ground truth from multiple noisy multivalued labels but did not account for image difficulty. Welinder and Perona [45] extended this model to other types of annotation, estimating annotator reliabilities and deciding the number of labels to request (and from which Turkers) in an online fashion.

Whitehill et al. [46] proposed a probabilistic model of the labeling process that simultaneously estimates the true label, item difficulty, and labeler expertise in an unsupervised manner. Using their GLAD algorithm, they demonstrated robustness to noisy and adversarial labelers as well as higher accuracy in inferring labels than a majority vote heuristic. However, their model does not consider annotator bias.

The method introduced by Welinder et al. [43] generalizes the GLAD algorithm by modeling annotator competence in broader terms. In the following sections, we examine their proposed model for image formation and the human annotation process, focusing on the binary image labeling case.

4.3 Problem Definition

The Welinder model [43] aims to capture various aspects of annotators and the labeling process. These aspects include:

- *Competency* - Accuracy and precision in labeling
- *Expertise* - Certain annotators may provide more reliable labels on different subsets of images, based on their areas of strength
- *Bias* - How one weighs errors varies annotator to annotator; for example, an optimistic annotator will accept some false positives and maintain an higher overall detection rate
- *Difficulty* - A difficult and/or ambiguous image may be challenging for annotators to label consistently

We now describe how the model represents these dimensions. In the annotation process, N images are produced. A variable z_i encapsulates the objects that are produced in image I_i. In the binary case, for example, $z_i \in \{1, 0\}$ can represent the presence/absence of a specific species of bird. Each image undergoes a visual transformation in a stochastic process that converts pixels into a vector x_i. This can be thought of as a vector of visual attributes (e.g. belly color, beak shape, head pattern, etc.) that represents factors an annotator can take into account in deciding a label.

Labeling is modeled in a signal detection framework. Out of M total annotators, a subset of annotators $\mathcal{J}_i$ labels image I_i. An annotator $j \in \mathcal{J}_i$ who labels I_i has access to a noisy signal $y_{ij} = x_i + n_{ij}$, where n_{ij} represents annotator-specific/image-specific differences such as visual acuity, attention, etc. and is parameterized by σ_j. Competent annotators will exhibit lower variance in noise n_{ij}.

The vector y_{ij} encodes the major components influencing an annotator's judgment in an annotation task. The unit vector $\hat{w}_j$ parametrizes an individual annotator, modeling his or her weighting on these components and representing annotator training or expertise. In a binary annotation task, an annotator assigns label $l_{ij} = 1$ if the scalar projection $\langle y_{ij}, \hat{w}_j \rangle$ is above a threshold $\hat{\tau}_j$, otherwise, $l_{ij} = 0$.

4.4 Model and Inference

The assumptions discussed in the previous section are combined to form a generative Bayesian model [43]. We proceed to discuss additional assumptions on the probability distributions that simplify inference.

We assume that the image representation x_i is generated by a Gaussian mixture model and normally distributed with variance θ_z; z_i specifies the component in the mixture model and is assumed to have Bernoulli prior distribution where $p(z_i = 1) = \beta$. In the binary annotation case, $z_i, l_{ij} \in \{0, 1\}$. Annotators are assumed to assign label l_{ij} according to a linear classifier, parametrized by a direction $\hat{w}_j$ of a decision plane and a bias $\hat{\tau}_j$; these are reparametrized as $w_j = \hat{w}_j/\sigma_j$ and $\tau_j = \hat{\tau}_j/\sigma_j$ to remove the constraint on $\hat{w}_j$ being a direction.

The simplified joint probability distribution can be written as follows:

$$p(\mathcal{L}, x, w, \tau) = \prod_{j=1}^{M} p(\tau_j|\gamma)p(w_j|\alpha)\times \prod_{i=1}^{N}\left(p(x_i|\theta_z, \beta)\sum_{j\in\mathcal{J}_i} p(l_{ij}|x_i, w_j, \tau_j)\right), \tag{1}$$

where $\mathcal{L}$ denotes the set of labels $\mathcal{L} = \{l_{ij}\}$, which comprises the only observed variables in this model. Since we are using prior distributions, we can apply MAP estimation and find optimal parameters by maximizing the posterior:

$$\begin{aligned}(x^*, w^*, \tau^*) &= \arg\max_{x,w,\tau} p(x, w, \tau|\mathcal{L}) \\ &= \arg\max_{x,w,\tau} m(x, w, \tau).\end{aligned} \tag{2}$$

where $m(x, w, \tau) = \log p(\mathcal{L}, x, w, \tau)$. To perform inference, we optimize $m(x, w, \tau)$, which is done with gradient ascent.

4.5 Worker "Schools of Thought"

Welinder et al. [43] are able to identify annotators based on the different attributes, represented by x_i, that they look for in images. We consider an example task to label presence in an image of a certain bird species, e.g. a duck, with value 1. Images may contain: (1) no birds, (2)

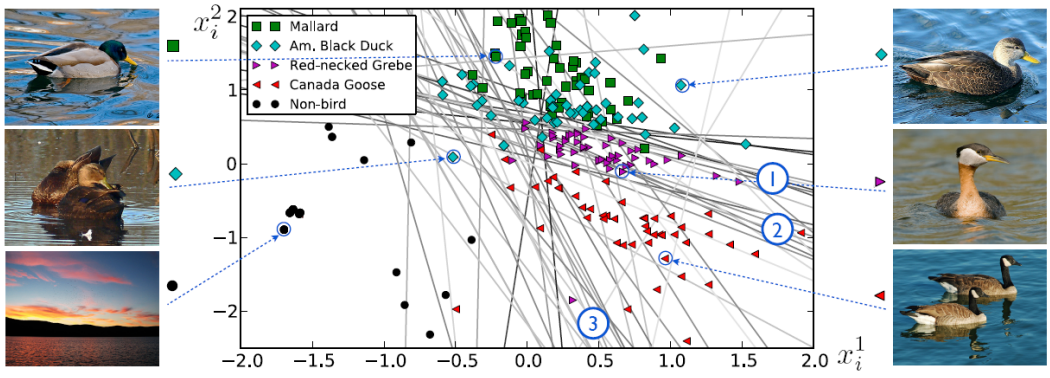

Figure 3: We observe the estimated x_i parameters for each image in a Waterbirds dataset, marked with symbols designating image class. The arrows point out x_i coordinates for certain example images. (source: [43]).

ducks, or (3) similar species to ducks such as grebes and geese (see Figure 3).

Results suggest that annotators use attributes in the image differently. Differing groups of annotators are able to separate: (1) ducks from all other images; (2) ducks and grebes from all other images; and (3) ducks, grebes, and geese from all other images (see corresponding circled numbers in Figure 3). Some annotators are more aware of the distinction between ducks and geese, while other annotators can differentiate between ducks and grebes, the latter of which have shorter necks and tend to look more duck-like than geese. The proposed model is able to distinguish between annotators with varying image feature/attribute preferences; in essence, it identifies different "schools of thought" among annotators.

4.6 Discussion

The Welinder model [43] captures the multidimensionality of annotators and the annotation process, and demonstrates promising results on various binary image labeling tasks. It is yet to be seen how well the model can be extended for more complex tasks with continuous annotations (e.g. segmentations or bounding boxes), as there may be additional underlying variables–for example in capturing human behavior or judgment–that have not been adequately taken into account.

Overall, different kinds of tasks may require different usages or combinations of reviewing strategies [4]. For example, using a combination of gold standard questions and a majority vote may provide sufficiently good results with minimal effort for binary decision tasks; with some additional effort, one can obtain improved results by applying the Welinder model. Neither of these approaches, however, would be applicable to open-ended tasks. Determining the optimal quality management strategy for a given task is an open problem.

It is important to note that predicting worker performance and label quality goes hand in hand with reducing the total number of labels and thus overall labeling cost. The more accurately one can estimate how difficult a task is or how well an annotator has labeled an image, the more confidently one can determine the number of additional labels needed, rather than blindly requesting a uniform maximum number of labels. Further discussion of approaches that address active label selection for computer vision applications is reserved for Section 5.

5 Cost Effective Strategies for Obtaining Labels

Obtaining large quantities of accurately labeled images on MTurk is generally a costly endeavor. One option is to select the most accurate and reliable Turkers to provide annotations [45, 42]. Alternatively, one can determine in

an online manner the number of assignments to request for a certain image, based on the current set of labels and the corresponding annotators [45].

If the goal is not only to obtain labeled data efficiently but also to learn a classifier from the data, then the problem falls in the domain of active learning. Traditional active learning approaches aim to minimize the number of labeled examples needed for learning; however, in a realistic image labeling context, there are multiple object labels per image and multiple types of annotation that can be obtained. The objective is now to request the most promising annotations and use those to update the current classifier. This involves trading off between the informativeness of a particular label and the manual effort required to obtain it.

Vijayanarasimhan and Grauman [35] note that the optimal use of manual effort may involve labels at different granularity levels. We discuss below how they formulate the problem in a multiple instance learning setting and are able to predict both cost and informativeness of different annotations. The proposed active learner is able to choose what image to label as well as what type of annotation to request: complete image segmentation, single object segmentation, or image-level category label.

5.1 Problem Definition

First, we discuss how the problem is posed in a multiple instance multi-label (MIML) framework. In standard multiple instance learning [8], a learner is given bags of instances. A positive bag contains at least one positive example, while none of the members of a negative bag are positive. In the MIML setting [35], each instance in a bag can be associated with one of C class labels, so a bag can be associated with multiple labels. We denote $\mathbb{L} = \{1, \ldots, C\}$ as the set of all possible class labels.

We define $\{(X_1, L_1), (X_2, L_2), \ldots, (X_N, L_N)\}$ as a set of training bags and their associated labels. Each bag contains a set of instances $X_i = \{x_1^i, x_2^i, \ldots, x_{n_i}^i\}$ and a set of labels $L_i = \{l_1^i, l_2^i, \ldots, l_{m_i}^i\}$; n_i denotes the number of instances in X_i and m_i is the number of labels in L_i. Usually, a bag has fewer unique labels than instances: $m_i \leq n_i$. In the visual categorization setting, an image represents a bag and instances are oversegmented regions, found automatically with a segmentation algorithm (Figure 4). An instance label names the object in a region, which is described by a feature vector.

The multi-class problem is decomposed into many binary problems, similar to one-vs-one classification. For each binary problem, an SVM is trained to discriminate bags containing label l_i from those containing $l_j, \forall i, j$. A multi-label set kernel [35] is used to weigh the feature vectors of instances within the bag, based on the estimated probability that the instance belongs to the class.

5.2 Prediction of Cost

Because images can be associated with multiple labels–class labels, object boundary segmentation, region/part labels, etc.–an active learning method must evaluate the value of an annotation for an image that can contain any number of unknown categories. These annotations require varying amounts of effort (and time) to obtain; for example, completely segmenting an image and labeling all the regions is a much more costly task to perform than providing a label for object presence.

Our goal is to accurately predict annotation time based solely on image content. This is done with supervised learning in order to estimate how difficult it is to segment an image. Training data is collected on MTurk, and workers are paid to segment images and name objects in segmented regions. The label for a training image is the average time taken by Turkers to complete a full annotation (Figure 4(d)). An SVM is then trained to predict the amount of manual effort required to annotate a particular image. This is used to build a cost function $\mathcal{C}(\mathbf{z})$ that takes candidate annotation $\mathbf{z}$ as input and returns the predicted time in seconds.

5.3 Prediction of Informativeness

In order to actively select candidate annotations, Vijayanarasimhan and Grauman [35] propose a measure based on the notion of value of information (VOI) to gauge the relative risk reduction provided by a new multi-label annotation. For each candidate annotation, a selection function evaluates its expected informativeness and subtracts out cost (Section 5.2). The goal is to predict the combination of image and annotation type that will produce the greatest decrease in risk for the current classifier, when penalized by the amount of manual effort required.

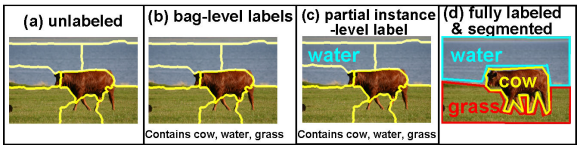

Figure 4: Vijayanarasimhan and Grauman [35] propose the MIML framework, in which images are represented as multi-label bags of regions or instances. (a) Oversegmented regions. (b) *Bag-level labels*: we know which categories are present in the image but we do not know in which regions. (c) *Partial instance-level labels*: some regions are labeled. (d) *Full annotation*: all regions are labeled and we have true object boundaries (source: [35]).

The risk terms are defined based on level of completeness in labeling. At any stage in the learning process, there exist distinct pools within the dataset: $\mathcal{X}_U$, the set of unlabeled examples (bags and instances); $\mathcal{X}_L$, the set of completely labeled examples; and $\mathcal{X}_P$, the set of partially labeled examples, which consists of bags with only a partial set of bag-level labels. The risk terms associated with these respective subsets of the dataset are:

$$\mathcal{R}(\mathcal{X}_L) = \sum_{X_i \in \mathcal{X}_L} \sum_{l \in L_i} r_l (1 - p(l|X_i)) \tag{3}$$

$$\mathcal{R}(\mathcal{X}_U) = \sum_{X_i \in \mathcal{X}_U} \sum_{l=1}^{C} r_l (1 - p(l|X_i)) \Pr(l|X_i) \tag{4}$$

$$\mathcal{R}(\mathcal{X}_P) = \sum_{X_i \in \mathcal{X}_P} \sum_{l \in L_i} r_l (1 - p(l|X_i)) + \sum_{l \in U_i} r_l (1 - p(l|X_i)) p(l|X_i), \tag{5}$$

where r_l represents the risk associated with misclassifying an example from class l; $p(l|X_i)$ is the probability X_i is classified with label l; $\Pr(l|X_i)$ is the true probability that unlabeled example X_i has label l (approximated as $\Pr(l|X_i) \approx p(l|X_i)$); and $U_i = \mathbb{L} \setminus L_i$. The reader is directed to [35] for more detail on these risk terms. It follows that the value of information for a candidate annotation $\mathbf{z}$ is:

$$VOI(\mathbf{z}) = \mathcal{R}(\mathcal{X}_L) + \mathcal{R}(\mathcal{X}_U) + \mathcal{R}(\mathcal{X}_P) - \left(\mathcal{R}(\hat{\mathcal{X}}_L) + \mathcal{R}(\hat{\mathcal{X}}_U) + \mathcal{R}(\hat{\mathcal{X}}_P)\right) - \mathcal{C}(\mathbf{z}), \tag{6}$$

where $\mathcal{R}(\hat{\mathcal{X}}_L)$, $\mathcal{R}(\hat{\mathcal{X}}_U)$, and $\mathcal{R}(\hat{\mathcal{X}}_P)$ denote the respective sets of labeled, unlabeled, and partially labeled data after obtaining annotation $\mathbf{z}$. A high VOI is desirable as it indicates that the total cost would decrease if a certain annotation were added.

As the VOI function depends on estimates for the risk of data that is yet to be labeled, the total risk is estimated by using the expected value: $\mathcal{R}(\hat{\mathcal{X}}_L)+\mathcal{R}(\hat{\mathcal{X}}_U)+\mathcal{R}(\hat{\mathcal{X}}_P) \approx \mathrm{E}[\mathcal{R}(\hat{\mathcal{X}}_L) + \mathcal{R}(\hat{\mathcal{X}}_U) + \mathcal{R}(\hat{\mathcal{X}}_P)]$.

5.4 Discussion

By posing the problem in a multiple instance learning setting, Vijayanarasimhan and Grauman [35] are able to formulate an active learning framework that actively selects image annotations, maximizing the expected benefit while considering the manual effort required. The MIML classifier is trained initially on a small set of labeled images. The active learner selects the label/example with maximal VOI from the set of partially labeled and unlabeled images, the classifier is then updated with this added label, and the process repeats. This framework enables one to study additional levels of supervision, for instance scene layout or part labels, and can potentially be extended to domains in computer vision outside of visual categorization.

The approach discussed above involves simultaneously minimizing cost while maximizing quality. An alternate approach to cost management is to place a limit on the total cost and explore the quality of results that can be obtained within that budget. Vijayanarasimhan et al. [36] investigated active batch selection of labels while satisfying a temporal budget constraint quantifying manual annotation cost. In addition, Vondrick et al. [41] examined the trade off in monetary cost between automation (quantified as the cost of CPU usage) and manual labeling, as a function of the difficulty of the labeling task.

6 Applications in Computer Vision

The most common usage of MTurk for computer vision-related applications is massive data collection of thousands, if not millions, of image labels. Recently, researchers have investigated how crowdsourcing can be used to supplement machine vision algorithms as well as how it can serve as a tool for better understanding components of computer vision. We present various examples of recent work in computer vision that leverage crowdsourcing.

6.1 Large-Scale Data Collection

In this section, we study examples of large-scale data collection for both images and video and how they utilize crowdsourcing.

6.1.1 Image Annotation

As mentioned previously, crowdsourcing enables one to label thousands of images for relatively low cost. This is advantageous in the creation of large-scale datasets, such as ImageNet [7], an image database of over 11 million images and counting, organized based on nouns in the WordNet hierarchy. Concepts in WordNet are described in phrases called "synonym sets" or synsets. Candidate images for ImageNet are harvested from the web and MTurk is used to verify each image for a given synset.

In terms of quality control, one must consider human error and disagreement, especially for more confusing synsets. Deng et al. [7] developed a simple algorithm that dynamically determines the number of agreements needed for different image categories. Responses from 10 users are collected on a subset of images for each synset, in order to establish a confidence score – a natural measure of "semantic difficulty"– indicating the probability of each image being correctly categorized. The remaining images in the synset are sent to MTurk for labeling until a confidence score threshold is attained. For certain synsets, users fail to reach consensus, suggesting that visualization of those concepts is challenging.

CUB-200 [44] is another example of a dataset that employed Turkers for image annotation. Intended for subordinate category classification, the dataset consists of over $6,000$ images of birds belonging to 200 species. Images are annotated with a rough segmentation, a bounding box, and binary attribute annotations. Subordinate categorization tasks are difficult for humans as well as computers (for example, accurately classifying bird species, car make/model, etc.) and CUB-200 aims to facilitate research in that area.

For a particular bird species, workers were shown an exemplar image and the corresponding Wikipedia entry for the species, and they were asked to rank the similarity between images and the exemplar using the options "same," "similar," "different," and "difficult" (chosen if occlusion or scale variation made comparison difficult). Images labeled as "same" or "similar" to the exemplar were kept. Workers were then requested to draw a rough segmentation of each bird, and bounding boxes were deduced from the rough outlines. A total of 5 Turkers were also asked to provide attribute annotations for each of the 25 visual attributes, harvested from an online bird field guide, as well as the confidence of their label: "definitely," "probably," or "guessing."

6.1.2 Video Annotation

Video annotation is one domain that especially requires cost-aware methods for labeling. The sheer quantity of frames necessitates clever ways of propagating annotations from a subset of keyframes. Yuen et al. [49] created the LabelMe video database using an online annotation tool similar to the original LabelMe interface [30] that allows users to draw polygons around objects for keyframes. In an offline preprocessing stage, camera motion is estimated as a homographic transformation. The web client then propagates the polygon label location temporally, while taking into account the camera parameters. Their interpolation framework assumes linear motion of constant velocity.

Vondrick et al. [41] address less predictable nonlinear motion in video by using a dynamic-programming based interpolation algorithm that tries to track the object between keyframes. The visual model of the tracked object is learned with a discriminative classifier trained to fire on positive bounding boxes and produce low scores on negatives.

Their annotation GUI is rendered by the client in real-time, and multiple workers can label the same sequence

simultaneously; their labels are merged into the data stream, so subsequent workers labeling the same video sequence see progressively more densely labeled video as the confident degenerate annotations are visually added to the sequence. A worker can then realize that an entity is already labeled and choose to label something else instead. Labeling is complete when multiple workers agree that there are no more entities left to label.

This labeling scheme introduces diversity across workers who are shown a random sequence, so a single worker cannot completely annotate one sequence. Additionally, if an adversarial worker provides poor annotations or demonstrates systematic bias, his/her performance will not affect all entities in a sequence.

6.2 Computer Vision with Humans in the Loop

A developing branch of research has focused on how humans can be added to different stages of a computer vision pipeline. They can be employed during training in an active learning setting [35], or during classification [3], as well as substituted for other machine vision components (detection, recognition, feature extraction, spatial modeling, etc.) [26, 25, 27]. These human-in-the-loop approaches serve various purposes, such as investigating deficiencies in computer vision or performing classification tasks that are difficult for computers.

Branson et al. [3] presented a "visual 20 questions game," an example of a subordinate category classification system that supports interaction between machine vision, human users, and experts. Given an input image, the computer queries the user and attempts to identify the true object class, while minimizing the total number of questions asked. As subordinate classification tasks are very challenging for humans without previous domain knowledge or expertise, the system poses questions about basic visual attributes than non-experts are able to answer. This hybrid human-computer framework enables users to drive up performance while minimizing the amount of human effort required. As computer vision algorithms improve, reliance on humans can be gradually reduced until humans are no longer needed at all. Branson et al. demonstrated their system on the CUB-200 dataset [44].

In addition, researchers have investigated using the crowd to advance knowledge about computer vision. Parikh and Zitnick [26] explored the role of features, training data, and algorithms in visual recognition tasks. They observed that the choice of features has the largest impact on machine recognition accuracy. In addition, they hypothesize that the feature representation used by humans is the primary factor giving them an advantage over computers; this is reasonable as there is evidence suggesting that humans can adaptively rely on different sets of features at test time, and the human feature representation is tuned for high performance at a variety of tasks.

Similar work [27] involved substituting humans for various combinations of components of a part-based person detector. Their findings suggest that part detection is the "weakest link," and substituting in human-detected parts significantly improves overall detector performance. Parikh and Grauman [25] also employed human annotators for building a nameable and discriminative attribute vocabulary that involves automatically identifying class confusions, having Turkers name the attribute that distinguishes the classes, and then incorporating Turker responses.

7 Conclusion

As use of crowdsourcing has dramatically increased with the advent of Amazon Mechanical Turk, it is clear that this distributed production model has its advantages. It provides a clear, feasible model for doing business and is relatively affordable. MTurk is particularly well-suited for tasks that benefit from human intelligence, reinforcing the "Turk philosophy": human computation is cheap and subsumes automated methods [41]. Methods exist for sufficient cost and quality management.

However, certain ethical issues have emerged, including what constitutes adequate compensation; requesters can potentially take advantage of workers by submitting them to unfair payment schemes [13]. Other issues arise with respect to the types of tasks that are posted on MTurk; a recent study estimated that more than 40% of the tasks listed on MTurk are spam [17], which include creating fake email/social networking accounts and ads, generating fake clicks, etc.

There is also a possible lack of diversity in participants, who self-select the tasks they perform [2, 21]. This sur-

vey has focused on crowdsourcing on MTurk, and experiments conducted through this venue may not generalize to other crowdsourcing marketplaces, given that the Turker population is not necessarily representative of population at large [23].

One may wonder, given these circumstances, what kind of tasks are best suited for MTurk deployment? While Turkers may not be identified experts, they can be successfully trained to perform more complex tasks. Their primary motivations are minimizing time spent and maximizing reward, which may conflict with overly complicated annotation schemes or requests and result in inattentiveness in labeling [9]. These concerns can conceivably be overcome with modifications to the user interface or task parameters; mediator services that target certain verticals (video transcription, translation, photo tagging, etc.) exist for this purpose [18].

From a computer vision perspective, crowdsourcing as actualized by MTurk has become an attractive paradigm for production, by having humans complete micro-tasks that would otherwise be very difficult for computers. The immaturity of the field may still be evident and the community will benefit from further study in certain areas. This includes investigating suitable design patterns and labeling interfaces for relevant tasks that yield the highest quality while minimizing cost.

Moreover, we will need to examine how to optimally incorporate a distributed workforce into a computer vision pipeline. Which tasks should we focus on developing algorithms for and which are more appropriate for the crowd [10]? The division of automated and human labor itself is highly dynamic, and it will constantly shift as machine vision algorithms improve and less reliance on humans is needed for certain tasks [3, 28].

These observations suggest future research directions involving hybrid approaches that combine crowdsourcing and computer vision to maximal effectiveness, or methods that enable finer control over quality, cost, and the speed at which results can be obtained. Some preliminary progress has been made in these areas [3, 29]. Additionally, the crowd can be used to validate these approaches and provide large-scale subjective evaluation and feedback.

By greatly facilitating large-scale image labeling, crowdsourcing has democratized the data collection process, eliminating dependence on stagnant datasets that potentially suffer from overtraining or overfitting [11]. Researchers are able to obtain image labels at more levels of granularity and develop more sophisticated algorithms. However, this on-demand mentality of accessible data collection may encourage datasets that are purposely framed to accommodate certain agendas, and standardization of crowdsourcing methods and parameters is needed.

It is yet to be seen what kind of enduring impact crowdsourcing will have on the area of computer vision. In the meantime, MTurk supports a readily available and willing workforce that has greatly facilitated data collection endeavors and continues to enable researchers to explore problems in computer vision at massive scales.

8 Acknowledgments

Thanks to Serge Belongie, Charles Elkan, and Lawrence Saul for serving on my research exam committee and for their helpful comments in the preparation of this document.

References

[1] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich. Soylent: a word processor with a crowd inside. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST, pages 313–322, 2010. 5, 6

[2] D. C. Brabham. Crowdsourcing as a model for problem solving. *Convergence: The International Journal of Research into New Media Technologies*, 14(1):75–90, 2008. 12

[3] S. Branson, C. Wah, B. Babenko, F. Schroff, P. Welinder, P. Perona, and S. Belongie. Visual recognition with humans in the loop. In *European Conference on Computer Vision (ECCV)*, Heraklion, Crete, Sept. 2010. 2, 12, 13

[4] J. Chen, N. Menezes, and A. Bradley. Opportunities for Crowdsourcing Research on Amazon Mechanical Turk. In *CHI 2011 Workshop on Crowdsourcing and Human Computation*, 2011. 3, 6, 8

[5] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popovic, and Foldit Players. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, August 2010. 3

[6] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied Statistics*, 28(1):20–28, 1979. 6

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 11

[8] T. G. Dietterich and R. H. Lathrop. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1997. 9

[9] I. Endres, A. Farhadi, D. Hoiem, and D. Forsyth. The benefits and challenges of collecting richer object annotations. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 1 –8, june 2010. 13

[10] D. Goldman and J. Brandt. Task Decomposition and Human Computation in Graphics and Vision. In *CHI 2011 Workshop on Crowdsourcing and Human Computation*, 2011. 5, 13

[11] V. Hester. Massive multiplayer human computation for fun, money and survival. http://crowdresearch.org/blog/?p=864, May 2011. 13

[12] L. Hoffmann. Crowd control. *Commun. ACM*, 52:16–17, March 2009. 1

[13] J. J. Horton and L. B. Chilton. The labor economics of paid crowdsourcing. In *Proceedings of the 11th ACM conference on Electronic commerce*, EC '10, pages 209–218, 2010. 3, 12

[14] J. Howe. The rise of crowdsourcing. *Wired Magazine*, 14(6), 06 2006. 1

[15] J. Howe. *Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business*. Three Rivers Press, 2009. 1

[16] E. Huang, H. Zhang, D. C. Parkes, K. Z. Gajos, and Y. Chen. Toward automatic task design: a progress report. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, pages 77–85, 2010. 4, 6

[17] P. Ipeirotis. Mechanical turk: Now with 40.92% spam. http://behind-the-enemy-lines.blogspot.com/2010/12/mechanical-turk-now-with-4092-spam.html, December 2010. 12

[18] P. Ipeirotis. Crowdsourcing goes professional: The rise of the verticals. http://behind-the-enemy-lines.blogspot.com/2011/03/crowdsourcing-goes-professional-rise-of.html, March 2011. 13

[19] P. G. Ipeirotis. Analyzing the amazon mechanical turk marketplace. *XRDS*, 17:16–21, December 2010. 3

[20] P. G. Ipeirotis, F. Provost, and J. Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, pages 64–67, 2010. 6

[21] A. Kittur, E. H. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 453–456, 2008. 4, 12

[22] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller. TurKit: Human computation algorithms on Mechanical Turk. In *UIST*, 2010. 4, 6

[23] W. Mason and D. J. Watts. Financial incentives and the "performance of crowds". In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '09, pages 77–85, 2009. 3, 13

[24] S. Nowak and S. Rüger. How reliable are annotations via crowdsourcing: a study about inter-annotator agreement for multi-label image annotation. In *Proceedings of the international conference on Multimedia information retrieval*, MIR '10, pages 557–566, 2010. 6

[25] D. Parikh and K. Grauman. Interactively Building a Discriminative Vocabulary of Nameable Attributes. In *To appear Computer Vision and Pattern Recognition*, 2011. 12

[26] D. Parikh and C. Zitnick. The role of features, algorithms and data in visual recognition. In *Computer Vision and Pattern Recognition*, 2010. 12

[27] D. Parikh and C. Zitnick. Finding the Weakest Link in Person Detectors. In *To appear Computer Vision and Pattern Recognition*, 2011. 12

[28] P. Perona. Vision of a visipedia. *Proceedings of the IEEE*, 98(8):1526 –1534, aug. 2010. 2, 13

[29] A. Quinn, B. Bederson, T. Yeh, and J. Lin. Crowdflow: Integrating machine learning with mechanical turk for speed-cost-quality flexibility. Technical Report HCIL-2010-09, University of Maryland, College Park, 2010. 13

[30] B. C. Russell, A. B. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173, 2008. 2, 11

[31] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 614–622, 2008. 6

[32] P. Smyth, U. M. Fayyad, M. C. Burl, P. Perona, and P. Baldi. Inferring ground truth from subjective labelling of Venus images. In *Neural Information Processing Systems*, pages 1085–1092, 1994. 6

[33] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 254–263, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. 6

[34] A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1 –8, june 2008. 1

[35] S. Vijayanarasimhan and K. Grauman. Cost-sensitive active visual category learning. *IJCV*, 91:24–44, January 2011. 9, 10, 12

[36] S. Vijayanarasimhan, P. Jain, and K. Grauman. Far-sighted active learning on a budget for image and video recognition. In *Computer Vision and Pattern Recognition*, pages 3035–3042, 2010. 10

[37] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *CHI '04: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 319–326, 2004. 2, 6

[38] L. von Ahn, S. Ginosar, M. Kedia, R. Liu, and M. Blum. Improving accessibility of the web with a computer game. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 79–82, 2006. 2

[39] L. von Ahn, R. Liu, and M. Blum. Peekaboom: a game for locating objects in images. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 55–64, 2006. 2

[40] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science*, 321(5895):1465–1468, 2008. 2

[41] C. Vondrick, D. Ramanan, and D. Patterson. Efficiently scaling up video annotation with crowdsourced marketplaces. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision ECCV 2010*, volume 6314 of *Lecture Notes in Computer Science*, pages 610–623. Springer Berlin / Heidelberg, 2010. 10, 11, 12

[42] P. Wais, S. Lingamneni, D. Cook, J. Fennell, B. Goldenberg, D. Lubarov, D. Marin, and H. Simons. Towards Building a High-Quality Workforce with Mechanical Turk. In *NIPS Workshop on Computational Social Science and the Wisdom of Crowds*, December 2010. 8

[43] P. Welinder, S. Branson, S. Belongie, and P. Perona. The multidimensional wisdom of crowds. In *Neural Information Processing Systems Conference (NIPS)*, 2010. 6, 7, 8

[44] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-ucsd birds 200. Technical Report CNS-TR-201, Caltech, 2010. 11, 12

[45] P. Welinder and P. Perona. Online crowdsourcing: Rating annotators and obtaining cost-effective labels. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 25 –32, june 2010. 6, 8, 9

[46] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose Vote Should Count More: Optimal Integration of Labels from Labelers of Unknown Expertise. In *Neural Information Processing Systems (NIPS)*, 2009. 6

[47] D. Wightman. Crowdsourcing human-based computation. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, NordiCHI '10, pages 551–560, New York, NY, USA, 2010. ACM. 1

[48] A. Williams. Wikinomics. http://www.wikinomics.com/blog/index.php/2009/02/09/crowdsourcing-versus-citizen-science/, February 2009. 2

[49] J. Yuen, B. Russell, and A. Torralba. The role of features, algorithms and data in visual recognition. In *International Conference on Computer Vision*, 2009. 11