ANALYZING, IMPROVING, AND LEVERAGING CROWDSOURCED VISUAL KNOWLEDGE REPRESENTATIONS

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTERS OF SCIENCE

Kenji Hata
June 2017
© Copyright by Kenji Hata 2017
All Rights Reserved
I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Masters of Science.

(Fei-Fei Li) Principal Co-Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Masters of Science.

(Michael Bernstein) Principal Co-Advisor

Approved for the University Committee on Graduate Studies

iii
Acknowledgements

First and foremost, I would like to express my utmost appreciation to my advisors Fei-Fei Li and Michael Bernstein for both nurturing my growth as a researcher and believing in me throughout my time at Stanford. Their guidance greatly developed my own maturity as a researcher and as a person.

I would also like to thank Oussama Khatib and Allison Okamura for sparking my initial interest in research as an undergraduate at Stanford. I thank Silvio Savarese for his support and guidance throughout my research career. I thank Ranjay Krishna for mentoring me throughout my Master’s program at Stanford.

Next, I would like to thank my co-authors, whom I have had the greatest joy working with and learning from. In alphabetical order, they are: Andrew Stanley, Allison Okamura, David Ayman Shamma, Frederic Ren, Joshua Kravitz, Juan Carlos Niebles, Justin Johnson, Li Fei-Fei, Michael Bernstein, Oliver Groth, Ranjay Krishna, Sherman Leung, Stephanie Chen, Yannis Kalantidis, and Yuke Zhu. More broadly, I give my appreciation to members of the Stanford Vision and Learning Lab and the Stanford HCI Lab, whose helpful discussions helped propel these works forward.

Finally, I would like to thank my parents, family, and friends for always believing in me throughout every step in life. It has been a great ride so far, and I cannot wait for what is next.
## Contents

<table>
<thead>
<tr>
<th>Acknowledgements</th>
<th>iv</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Thesis Outline</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Previously Published Papers</td>
<td>2</td>
</tr>
<tr>
<td><strong>2 Visual Genome</strong></td>
<td>3</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Related Work</td>
<td>9</td>
</tr>
<tr>
<td>2.2.1 Datasets</td>
<td>9</td>
</tr>
<tr>
<td>2.2.2 Image Descriptions</td>
<td>11</td>
</tr>
<tr>
<td>2.2.3 Objects</td>
<td>11</td>
</tr>
<tr>
<td>2.2.4 Attributes</td>
<td>12</td>
</tr>
<tr>
<td>2.2.5 Relationships</td>
<td>12</td>
</tr>
<tr>
<td>2.2.6 Question Answering</td>
<td>13</td>
</tr>
<tr>
<td>2.2.7 Knowledge Representation</td>
<td>13</td>
</tr>
<tr>
<td>2.3 Visual Genome Data Representation</td>
<td>14</td>
</tr>
<tr>
<td>2.3.1 Multiple regions and their descriptions</td>
<td>14</td>
</tr>
<tr>
<td>2.3.2 Multiple objects and their bounding boxes</td>
<td>15</td>
</tr>
<tr>
<td>2.3.3 A set of attributes</td>
<td>15</td>
</tr>
<tr>
<td>2.3.4 A set of relationships</td>
<td>16</td>
</tr>
<tr>
<td>2.3.5 A set of region graphs</td>
<td>16</td>
</tr>
<tr>
<td>2.3.6 One scene graph</td>
<td>17</td>
</tr>
<tr>
<td>2.3.7 A set of question answer pairs</td>
<td>17</td>
</tr>
<tr>
<td>2.4 Dataset Statistics and Analysis</td>
<td>18</td>
</tr>
<tr>
<td>2.4.1 Image Selection</td>
<td>19</td>
</tr>
<tr>
<td>2.4.2 Region Description Statistics</td>
<td>20</td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Section</th>
<th>Table Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>A comparison of existing datasets with Visual Genome. We show that Visual Genome</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>has an order of magnitude more descriptions and question answers. It also has a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>more diverse set of object, attribute, and relationship classes. Additionally,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Visual Genome contains a higher density of these annotations per image. The number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>of distinct categories in Visual Genome are calculated by lower-casing and stemming</td>
<td></td>
</tr>
<tr>
<td></td>
<td>names of objects, attributes and relationships.</td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>Comparison of Visual Genome objects and categories to related datasets.</td>
<td>24</td>
</tr>
<tr>
<td>2.3</td>
<td>The average number of objects, attributes, and relationships per region graph and</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>per scene graph.</td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>Precision, recall, and mapping accuracy percentages for object, attribute, and rel-</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>tionship canonicalization.</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>Geographic distribution of countries from where crowd workers contributed to Visual</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>Genome.</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>We compare the conventional approach for binary verification tasks (image verifica-</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>tion, sentiment analysis, word similarity and topic detection) with our technique</td>
<td></td>
</tr>
<tr>
<td></td>
<td>and compute precision and recall scores. Precision scores, recall scores and speedups</td>
<td></td>
</tr>
<tr>
<td></td>
<td>are calculated using 3 workers in the conventional setting. Image verification,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sentiment analysis and word similarity used 5 workers using our technique, while</td>
<td></td>
</tr>
<tr>
<td></td>
<td>topic detection used only 2 workers. We also show the time taken (in seconds) for 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>worker to do each task.</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>The number of workers, tasks, and annotations collected for image descriptions,</td>
<td>74</td>
</tr>
<tr>
<td></td>
<td>question answering, and verifications.</td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>Data collected from the verification experiment. A total of 1,134 workers were</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>divided up into four conditions, with a high or low threshold and transparency.</td>
<td></td>
</tr>
</tbody>
</table>
6.1 (First row) Results for the attribute prediction task where we only predict attributes for a given image crop. (Second row) Attribute-object prediction experiment where we predict both the attributes as well as the object from a given crop of the image.

6.2 Results for relationship classification (first row) and joint classification (second row) experiments.

6.3 Results for the region description generation experiment. Scores in the first row are for the region descriptions generated from the NeuralTalk model trained on Flickr8K, and those in the second row are for those generated by the model trained on Visual Genome data. BLEU, CIDEr, and METEOR scores all compare the predicted description to a ground truth in different ways.

6.4 Baseline QA performances in the 6 different question types. We report human evaluation as well as a baseline method that predicts the most frequently occurring answer in the dataset.

7.1 We report Bleu (B), METEOR (M) and CIDEr (C) captioning scores for the task of dense-captioning events. On the top table, we report performances of just our captioning module with ground truth proposals. On the bottom table, we report the combined performances of our complete model, with proposals predicted from our proposal module. Since prior work has focused only on describing entire videos and not also detecting a series of events, we only compare existing video captioning models using ground truth proposals.

7.2 We report the effects of context on captioning the 1st, 2nd and 3rd events in a video. We see that performance increases with the addition of past context in the online model and with future context in full model.

7.3 Results for video and paragraph retrieval. We see that the utilization of context to encode video events help us improve retrieval. R@k measures the recall at varying thresholds k and med. rank measures the median rank the retrieval.

7.4 Compared to other video datasets, ActivityNet Captions (ANC) contains long videos with a large number of sentences that are all temporally localized and is the only dataset that contains overlapping events. (Loc. shows which datasets contain temporally localized language descriptions. Bold fonts are used to highlight the nearest comparison of our model with existing models.)
List of Figures

2.1 An overview of the data needed to move from perceptual awareness to cognitive understanding of images. We present a dataset of images densely annotated with numerous region descriptions, objects, attributes, and relationships. Some examples of region descriptions (e.g. “girl feeding large elephant” and “a man taking a picture behind girl”) are shown (top). The objects (e.g. elephant), attributes (e.g. large) and relationships (e.g. feeding) are shown (bottom). Our dataset also contains image related question answer pairs (not shown).................. 4

2.2 An example image from the Visual Genome dataset. We show 3 region descriptions and their corresponding region graphs. We also show the connected scene graph collected by combining all of the image’s region graphs. The top region description is “a man and a woman sit on a park bench along a river.” It contains the objects: man, woman, bench and river. The relationships that connect these objects are: sits_on(man, bench), in_front_of(man, river), and sits_on(woman, bench)............ 6

2.3 An example image from our dataset along with its scene graph representation. The scene graph contains objects (child, instructor, helmet, etc.) that are localized in the image as bounding boxes (not shown). These objects also have attributes: large, green, behind, etc. Finally, objects are connected to each other through relationships: wears(child, helmet), wears(instructor, jacket), etc.................. 7

2.4 A representation of the Visual Genome dataset. Each image contains region descriptions that describe a localized portion of the image. We collect two types of question answer pairs (QAs): freeform QAs and region-based QAs. Each region is converted to a region graph representation of objects, attributes, and pairwise relationships. Finally, each of these region graphs are combined to form a scene graph with all the objects grounded to the image. Best viewed in color................................. 8
To describe all the contents of and interactions in an image, the Visual Genome dataset includes multiple human-generated image regions descriptions, with each region localized by a bounding box. Here, we show three regions descriptions on various image regions: “man jumping over a fire hydrant,” “yellow fire hydrant,” and “woman in shorts is standing behind the man.”

From all of the region descriptions, we extract all objects mentioned. For example, from the region description “man jumping over a fire hydrant,” we extract man and fire hydrant.

Some descriptions also provide attributes for objects. For example, the region description “yellow fire hydrant” adds that the fire hydrant is yellow. Here we show two attributes: yellow and standing.

Our dataset also captures the relationships and interactions between objects in our images. In this example, we show the relationship jumping over between the objects man and fire hydrant.

A distribution of the top 25 image synsets in the Visual Genome dataset. A variety of synsets are well represented in the dataset, with the top 25 synsets having at least 800 example images each. Note that an image synset is the label of the entire image according to the ImageNet ontology and are separate from the synsets for objects, attributes and relationships.

(a) An example image from the dataset with its region descriptions. We only display localizations for 6 of the 50 descriptions to avoid clutter; all 50 descriptions do have corresponding bounding boxes. (b) All 50 region bounding boxes visualized on the image.

A distribution of the width of the bounding box of a region description normalized by the image width. (b) A distribution of the height of the bounding box of a region description normalized by the image height.

A distribution of the number of words in a region description. The average number of words in a region description is 5, with shortest descriptions of 1 word and longest descriptions of 16 words.

The process used to convert a region description into a 300-dimensional vectorized representation.

(a) A plot of the most common visual concepts or phrases that occur in region descriptions. The most common phrases refer to universal visual concepts like “blue sky,” “green grass,” etc. (b) A plot of the most frequently used words in region descriptions. Each word is treated as an individual token regardless of which region description it came from. Colors occur the most frequently, followed by common objects like man and dog and universal visual concepts like “sky.”
2.15 (a) Example illustration showing four clusters of region descriptions and their overall themes. Other clusters not shown due to limited space. (b) Distribution of images over number of clusters represented in each image's region descriptions. (c) We take Visual Genome with 5 random descriptions taken from each image and MS-COCO dataset with all 5 sentence descriptions per image and compare how many clusters are represented in the descriptions. We show that Visual Genome’s descriptions are more varied for a given image, with an average of 4 clusters per image, while MS-COCO’s images have an average of 2 clusters per image.

2.16 (a) Distribution of the number of objects per region. Most regions have between 0 and 2 objects. (b) Distribution of the number of objects per image. Most images contain between 15 and 20 objects.

2.17 Comparison of object diversity between various datasets. Visual Genome far surpasses other datasets in terms of number of categories. When considering only the top 80 object categories, it contains a comparable number of objects as MS-COCO. The dashed line is a visual aid connecting the two Visual Genome data points.

2.18 (a) Examples of objects in Visual Genome. Each object is localized in its image with a tightly drawn bounding box. (b) Plot of the most frequently occurring objects in images. People are the most frequently occurring objects in our dataset, followed by common objects and visual elements like building, shirt, and sky.

2.19 Distribution of the number of attributes (a) per image, (b) per region description, (c) per object.

2.20 (a) Distribution showing the most common attributes in the dataset. Colors (e.g. white, red) and materials (e.g. wooden, metal) are the most common. (b) Distribution showing the number of attributes describing people. State-of-motion verbs (e.g. standing, walking) are the most common, while certain sports (e.g. skiing, surfing) are also highly represented due to an image source bias in our image set.

2.21 (a) Graph of the person-describing attributes with the most co-occurrences. Edge thickness represents the frequency of co-occurrence of the two nodes. (b) A subgraph showing the co-occurrences and intersections of three cliques, which appear to describe water (top right), hair (bottom right), and some type of animal (left). Edges between cliques have been removed for clarity.

2.22 Distribution of relationships (a) per image region, (b) per image object, (c) per image.

2.23 (a) A sample of the most frequent relationships in our dataset. In general, the most common relationships are spatial (on top of, on side of, etc.). (b) A sample of the most frequent relationships involving humans in our dataset. The relationships involving people tend to be more action oriented (walk, speak, run, etc.).
2.24 (a) Distribution of subjects for the relationship riding. Subjects comprise of people-like entities like person, man, policeman, boy, and skateboarder that can ride other objects. On the other hand, objects like horse, bike, elephant and motorcycle are entities that can afford riding.

(b) Distribution of objects for the relationship riding. Subjects comprise of people-like entities like person, man, policeman, boy, and skateboarder that can ride other objects. On the other hand, objects like horse, bike, elephant and motorcycle are entities that can afford riding.

2.25 Example QA pairs in the Visual Genome dataset. Our QA pairs cover a spectrum of visual tasks from recognition to high-level reasoning.

2.26 (a) Distribution of question types by starting words. This figure shows the distribution of the questions by their first three words. The angles of the regions are proportional to the number of pairs from the corresponding categories. We can see that “what” questions are the largest category with nearly half of the QA pairs. (b) Question and answer lengths by question type. The bars show the average question and answer lengths of each question type. The whiskers show the standard deviations. The factual questions, such as “what” and “how” questions, usually come with short answers of a single object or a number. This is only because “how” questions are disproportionately counting questions that start with “how many”. Questions from the “where” and “why” categories usually have phrases and sentences as answers.

2.27 An example image from the Visual Genome dataset with its region descriptions, QA pairs, objects, attributes, and relationships canonicalized. The large text boxes are WordNet synsets referenced by this image. For example, the carriage is mapped to carriage.n.02: a vehicle with wheels drawn by one or more horses. We do not show the bounding boxes for the objects in order to allow readers to see the image clearly. We also only show a subset of the scene graph for this image to avoid cluttering the figure.

2.28 Distribution of the 25 most common synsets mapped from the words and phrases extracted from region descriptions which represent objects in (a) region descriptions and question answers and (b) objects.

2.29 Distribution of the 25 most common synsets mapped from (a) attributes and (b) relationships.

3.1 (a) Age and (b) gender distribution of Visual Genome’s crowd workers.

3.2 Good (left) and bad (right) bounding boxes for the phrase “a street with a red car parked on the side,” judged on coverage.

3.3 Good (left) and bad (right) bounding boxes for the object fox, judged on both coverage as well as quality.

3.4 Each object (fox) has only one bounding box referring to it (left). Multiple boxes drawn for the same object (right) are combined together if they have a minimum threshold of 0.9 intersection over union.
4.1 (a) Images are shown to workers at 100ms per image. Workers react whenever they see a dog. (b) The true labels are the ground truth dog images. (c) The workers' keypresses are slow and occur several images after the dog images have already passed. We record these keypresses as the observed labels. (d) Our technique models each keypress as a delayed Gaussian to predict the probability of an image containing a dog from these observed labels.

4.2 (a) Task instructions inform workers that we expect them to make mistakes since the items will be displayed rapidly. (b) A string of countdown images prepares them for the rate at which items will be displayed. (c) An example image of a "dog" shown in the stream—the two images appearing behind it are included for clarity but are not displayed to workers. (d) When the worker presses a key, we show the last four images below the stream of images to indicate which images might have just been labeled.

4.3 Example raw worker outputs from our interface. Each image was displayed for 100ms and workers were asked to react whenever they saw images of "a person riding a motorcycle." Images are shown in the same order they appeared in for the worker. Positive images are shown with a blue bar below them and users' keypresses are shown as red bars below the image to which they reacted.

4.4 We plot the change in recall as we vary percentage of positive items in a task. We experiment at varying display speeds ranging from 100ms to 500ms. We find that recall is inversely proportional to the rate of positive stimuli and not to the percentage of positive items.

4.5 We study the precision (left) and recall (right) curves for detecting "dog" (top), a person on a motorcycle (middle) and "eating breakfast" (bottom) images with a redundancy ranging from 1 to 5. There are 500 ground truth positive images in each experiment. We find that our technique works for simple as well as hard concepts.

4.6 We study the effects of redundancy on recall by plotting precision and recall curves for detecting "a person on a motorcycle" images with a redundancy ranging from 1 to 10. We see diminishing increases in precision and recall as we increase redundancy. We manage to achieve the same precision and recall scores as the conventional approach with a redundancy of 10 while still achieving a speedup of 5.

4.7 Precision (left) and recall (right) curves for sentiment analysis (top), word similarity (middle) and topic detection (bottom) images with a redundancy ranging from 1 to 5. Vertical lines indicate the number of ground truth positive examples.

5.1 A distribution of the number of workers for each of the three datasets. A small number of persistent workers complete most of the work: the top 20% of workers completed roughly 90% of all tasks.
5.2 Self reported gender (left) and age distribution (right) of 298 workers who completed at least 100 of the image description, question answer, or binary verification tasks. 75

5.3 The average accuracy over the lifetime of each worker who completed over 100 tasks in each of the three datasets. The top row shows accuracy for image descriptions, the middle row shows accuracy for question answering, and the bottom row shows accuracy for the verification dataset. 76

5.4 A selection of individual workers’ accuracy over time during the question answering task. Each worker remains relatively constant throughout his or her entire lifetime. 77

5.5 On average, workers who repeatedly completed the image description (top row) or question answering (bottom row) tasks gave descriptions or questions with increasingly similar syntactic structures. 78

5.6 Image descriptions written by a satisficing worker on a task completed near the start of their lifetime (left) and their last completed task (right). Despite the images being visually similar, the phrases submitted in the last task are much less diverse than the ones submitted in the earlier task. 80

5.7 As workers gain familiarity with a task, they become faster. Verification tasks speed up by 25% from novice to experienced workers. 81

5.8 An example binary verification task where workers are asked to determine if the phrase “the zebras have stripes” is a factually correct description of the image region surrounded within the red box. There were 58 verification questions in each task. 82

5.9 Examples of attention checks placed in our binary verification tasks. Each attention check was designed such that they were easily identified as correct or incorrect. “An elephant’s trunk” (left) is a positive attention check while “A very tall sailboat” (right) is an incorrect attention check. We rated worker’s quality by measuring how well they performed on these attention checks. 83

5.10 Worker accuracy was unaffected by the threshold level and by the visibility of the threshold. The dotted black line indicates the threshold that the workers were supposed to adhere to. 85

5.11 It is possible to model a worker’s future quality by observing only a small glimpse of their initial work. Our all workers’ average baseline assumes that all workers perform similarly and manages an error in individual worker quality prediction of 6.9%. Meanwhile, by just observing the first 5 tasks, our average and sigmoid models achieve 3.4% and 3.7% prediction error respectively. As we observe more hits, the sigmoid model is able to represent workers better than the average model. 86
6.1 (a) Example predictions from the attribute prediction experiment. Attributes in the first row are predicted correctly, those in the second row differ from the ground truth but still correctly classify an attribute in the image, and those in the third row are classified incorrectly. The model tends to associate objects with attributes (e.g., elephant with grazing). (b) Example predictions from the joint object-attribute prediction experiment.

6.2 (a) Example predictions from the relationship prediction experiment. Relationships in the first row are predicted correctly, those in the second row differ from the ground truth but still correctly classify a relationship in the image, and those in the third row are classified incorrectly. The model learns to associate animals leaning towards the ground as eating or drinking and bikes with riding. (b) Example predictions from the relationship-objects prediction experiment. The figure is organized in the same way as Figure (a). The model is able to predict the salient features of the image but fails to distinguish between different objects (e.g., boy and woman and car and bus in the bottom row).

6.3 Example predictions from the region description generation experiment by a model trained on Visual Genome region descriptions. Regions in the first column (left) accurately describe the region, and those in the second column (right) are incorrect and unrelated to the corresponding region.

7.1 Dense-captioning events in a video involves detecting multiple events that occur in a video and describing each event using natural language. These events are temporally localized in the video with independent start and end times, resulting in some events that might also occur concurrently and overlap in time.

7.2 Complete pipeline for dense-captioning events in videos with descriptions. We first extract C3D features from the input video. These features are fed into our proposal module at varying stride to predict both short as well as long events. Each proposal, which consists of a unique start and end time and a hidden representation, is then used as input into the captioning module. Finally, this captioning model leverages context from neighboring events to generate each event description.

7.3 The parts of speech distribution of ActivityNet Captions compared with Visual Genome, a dataset with multiple sentence annotations per image. There are many more verbs and pronouns represented in ActivityNet Captions, as the descriptions often focus on actions.

7.4 Qualitative results of our dense captioning model.

7.5 Evaluating our proposal module, we find that sampling videos at varying strides does in fact improve the module’s ability to localize events, specially longer events.
7.6 (a) The number of sentences within paragraphs is normally distributed, with on average 3.65 sentences per paragraph. (b) The number of words per sentence within paragraphs is normally distributed, with on average 13.48 words per sentence.

7.7 Distribution of number of sentences with respect to video length. In general the longer the video the more sentences there are, so far on average each additional minute adds one more sentence to the paragraph.

7.8 Distribution of annotations in time in ActivityNet Captions videos, most of the annotated time intervals are closer to the middle of the videos than to the start and end.

7.9 (a) The most frequently used words in ActivityNet Captions with stop words removed. (b) The most frequently used bigrams in ActivityNet Captions.

7.10 (a) Interface when a worker is writing a paragraph. Workers are asked to write a paragraph in the text box and press "Done Writing Paragraph" before they can proceed with grounding each of the sentences. (b) Interface when labeling sentences with start and end timestamps. Workers select each sentence, adjust the range slider indicating which segment of the video that particular sentence is referring to. They then click save and proceed to the next sentence. (c) We show examples of good and bad annotations to workers. Each task contains one good and one bad example video with annotations. We also explain why the examples are considered to be good or bad.

7.11 More qualitative dense-captioning captions generated using our model. We show captions with the highest overlap with ground truth captions.

7.12 The number of videos (red) corresponding to each ActivityNet class label, as well as the number of videos (blue) that has the label appearing in their ActivityNet Captions paragraph descriptions.
Chapter 1

Introduction

1.1 Motivation

Despite recent breakthroughs in solving perceptual tasks like image classification, modern computer vision models are still unable to perform well on reasoning tasks such as captioning a scene or answering questions. A potential reason for the performance gap is that current computer vision models are often trained on traditional, large-scale datasets created for perceptual tasks. Therefore, as the complexity for problems in computer vision rises, so does the need for the creation and use of new, richer large-scale datasets.

Interesting problems in both human-computer interaction and computer vision arise in the creation of these datasets. For example, better understanding the crowdsourcing processes common in the creation of modern datasets may help reduce costs while simultaneously improving the quality of the data collected. Additionally, new methods for automating many parts of a crowdsourcing pipeline may leverage modern computer vision techniques.

Ultimately, the main goal of this thesis is two-fold. First, we want to understand and improve the crowdsourcing pipeline for collecting large-scale visual datasets. Second, we want to demonstrate how we can use these new computer vision datasets to build models that can better tackle more complex reasoning tasks.

1.2 Thesis Outline

In this thesis, we first focus on the theme of understanding the entire process of building computer vision models that leverage large-scale data. In Chapter 2, we introduce Visual Genome, the densest crowdsourced dataset for large-scale visual content. The concepts of connecting objects, attributes, and relationships within each image enable us to build scene graphs of images and form the densest database for visual knowledge representation. Chapter 3 covers the main crowdsourcing pipeline
we used to collect the Visual Genome dataset. We outline the lessons learned to transfer common strategies that may be employed in the collection of other computer vision or natural language processing datasets. Chapter 4 dives into a novel crowdsourcing method that models worker latency to rapidly collect binary labels for large-scale datasets like Visual Genome. This approach leads to an order of magnitude speedup in crowdwork, leading to significant cost reductions. Chapter 5 studies how we can use the collection of large datasets to better understand crowd workers at scale. We find that crowd workers maintain a consistent quality level during the completion of microtasks, allowing for dataset creators to easily ascertain good crowd workers early on. Chapter 6 discusses how we can leverage Visual Genome to solve new tasks in computer vision. Chapter 7 focuses on the collection and use of a new, large-scale video dataset in order to densely caption videos with sentences. In this chapter, we illustrate the complexity of tasks that can be achieved with the construction of new datasets with new computer vision models. Finally, Chapter 8 provides a brief summary of the future directions and applications of the research discussed.

1.3 Previously Published Papers

The majority of contributions of this thesis has previously appeared in various publications: 
118 (Chapters 2 and 3), 119 (Chapter 4), 81 (Chapter 5), 117 (Chapter 6). Other publications are out of context with the theme of this thesis and consequently are not included.
Chapter 2

Visual Genome

2.1 Introduction

A holy grail of computer vision is the complete understanding of visual scenes: a model that is able to name and detect objects, describe their attributes, and recognize their relationships. Understanding scenes would enable important applications such as image search, question answering, and robotic interactions. Much progress has been made in recent years towards this goal, including image classification [180, 219, 120, 231] and object detection [71, 212, 70, 190]. An important contributing factor is the availability of a large amount of data that drives the statistical models that underpin today’s advances in computational visual understanding. While the progress is exciting, we are still far from reaching the goal of comprehensive scene understanding. As Figure 2.1 shows, existing models would be able to detect discrete objects in a photo but would not be able to explain their interactions or the relationships between them. Such explanations tend to be cognitive in nature, integrating perceptual information into conclusions about the relationships between objects in a scene [19, 63]. A cognitive understanding of our visual world thus requires that we complement computers’ ability to detect objects with abilities to describe those objects [97] and understand their interactions within a scene [204].

There is an increasing effort to put together the next generation of datasets to serve as training and benchmarking datasets for these deeper, cognitive scene understanding and reasoning tasks, the most notable being MS-COCO [140] and VQA [2]. The MS-COCO dataset consists of 300K real-world photos collected from Flickr. For each image, there is pixel-level segmentation of 80 object classes (when present) and 5 independent, user-generated sentences describing the scene. VQA adds

Visual Genome was a highly collaborative project between many students, faculty, and industry affiliates. My main contributions to the Visual Genome project involved helping build the crowdsourcing framework, benchmarking the dataset with deep neural networks, and then iterating on the dataset to be usable for computer vision researchers.
CHAPTER 2. VISUAL GENOME

Figure 2.1: An overview of the data needed to move from perceptual awareness to cognitive understanding of images. We present a dataset of images densely annotated with numerous region descriptions, objects, attributes, and relationships. Some examples of region descriptions (e.g., “girl feeding large elephant” and “a man taking a picture behind girl”) are shown (top). The objects (e.g. elephant), attributes (e.g. large) and relationships (e.g. feeding) are shown (bottom). Our dataset also contains image related question answer pairs (not shown).

to this a set of 614K question answer pairs related to the visual contents of each image (see more details in Section 2.2.1). With this information, MS-COCO and VQA provide a fertile training and testing ground for models aimed at tasks for accurate object detection, segmentation, and summary-level image captioning [111, 150, 109] as well as basic QA [189, 147, 66, 146]. For example, a state-of-the-art model [109] provides a description of one MS-COCO image in Figure 2.1 as “two men are standing next to an elephant.” But what is missing is the further understanding of where each object is, what each person is doing, what the relationship between the person and elephant is, etc. Without such relationships, these models fail to differentiate this image from other images of people next to elephants.

To understand images thoroughly, we believe three key elements need to be added to existing datasets: a grounding of visual concepts to language [111], a more complete set of descriptions and QAs for each image based on multiple image regions [102], and a formalized representation of the components of an image [82]. In the spirit of mapping out this complete information of the visual world, we introduce the Visual Genome dataset. The first release of the Visual Genome dataset uses 108,077 images from the intersection of the YFCC100M [234] and MS-COCO [140]. Section 2.4 provides a more detailed description of the dataset. We highlight below the motivation and contributions of the three key elements that set Visual Genome apart from existing datasets.

The Visual Genome dataset regards relationships and attributes as first-class citizens of the annotation space, in addition to the traditional focus on objects. Recognition of relationships and attributes is an important part of the complete understanding of the visual scene, and in many cases,
these elements are key to the story of a scene (e.g., the difference between “a dog chasing a man” versus “a man chasing a dog”). The Visual Genome dataset is among the first to provide a detailed labeling of object interactions and attributes, **grounding visual concepts to language**.

An image is often a rich scenery that cannot be fully described in one summarizing sentence. The scene in Figure 2.1 contains multiple “stories”: “a man taking a photo of elephants,” “a woman feeding an elephant,” “a river in the background of lush grounds,” etc. Existing datasets such as Flickr 30K [275] and MS-COCO [140] focus on high-level descriptions of an image. Instead, for each image in the Visual Genome dataset, we collect more than 50 descriptions for different regions in the image, providing a much denser and more **complete set of descriptions of the scene**. In addition, inspired by VQA [2], we also collect an average of 17 question answer pairs based on the descriptions for each image. Region-based question answers can be used to jointly develop NLP and vision models that can answer questions from either the description or the image, or both of them.

With a set of dense descriptions of an image and the explicit correspondences between visual pixels (i.e. bounding boxes of objects) and textual descriptors (i.e. relationships, attributes), the Visual Genome dataset is poised to be the first image dataset that is capable of providing a structured **formalized representation** of an image, in the form that is widely used in knowledge base representations in NLP [281, 77, 38, 223]. For example, in Figure 2.1, we can formally express the relationship holding between the woman and food as holding(woman, food). Putting together all the objects and relations in a scene, we can represent each image as a scene graph [102]. The scene graph representation has been shown to improve semantic image retrieval [102, 210] and image captioning [57, 28, 78]. Furthermore, all objects, attributes and relationships in each image in the Visual Genome dataset are canonicalized to its corresponding WordNet [160] ID (called a synset ID). This mapping connects all images in Visual Genome and provides an effective way to consistently query the same concept (object, attribute, or relationship) in the dataset. It can also potentially help train models that can learn from contextual information from multiple images.

In this paper, we introduce the Visual Genome dataset with the aim of training and benchmarking the next generation of computer models for comprehensive scene understanding. The paper proceeds as follows: In Section 2.3, we provide a detailed description of each component of the dataset. Section 2.2 provides a literature review of related datasets as well as related recognition tasks. Section 3 discusses the crowdsourcing strategies we deployed in the ongoing effort of collecting this dataset. Section 2.4 is a collection of data analysis statistics, showcasing the key properties of the Visual Genome dataset. Last but not least, Section 6.1 provides a set of experimental results that use Visual Genome as a benchmark.

---

1The Lotus Hill Dataset [271] also provides a similar annotation of object relationships, see Sec 2.2.1.
2COCO has multiple sentences generated independently by different users, all focusing on providing an overall, one sentence description of the scene.
3Further visualizations, API, and additional information on the Visual Genome dataset can be found online: https://visualgenome.org
 CHAPTER 2. VISUAL GENOME

Figure 2.2: An example image from the Visual Genome dataset. We show 3 region descriptions and their corresponding region graphs. We also show the connected scene graph collected by combining all of the image’s region graphs. The top region description is “a man and a woman sit on a park bench along a river.” It contains the objects: man, woman, bench and river. The relationships that connect these objects are: sits_on(man, bench), in_front_of(man, river), and sits_on(woman, bench).
Figure 2.3: An example image from our dataset along with its scene graph representation. The scene graph contains objects (child, instructor, helmet, etc.) that are localized in the image as bounding boxes (not shown). These objects also have attributes: large, green, behind, etc. Finally, objects are connected to each other through relationships: wears(child, helmet), wears(instructor, jacket), etc.
CHAPTER 2. VISUAL GENOME

Figure 2.4: A representation of the Visual Genome dataset. Each image contains region descriptions that describe a localized portion of the image. We collect two types of question answer pairs (QAs): freeform QAs and region-based QAs. Each region is converted to a region graph representation of objects, attributes, and pairwise relationships. Finally, each of these region graphs are combined to form a scene graph with all the objects grounded to the image. *Best viewed in color*
2.2 Related Work

We discuss existing datasets that have been released and used by the vision community for classification and object detection. We also mention work that has improved object and attribute detection models. Then, we explore existing work that has utilized representations similar to our relationships between objects. In addition, we dive into literature related to cognitive tasks like image description, question answering, and knowledge representation.

2.2.1 Datasets

Datasets (Table 2.1) have been growing in size as researchers have begun tackling increasingly complicated problems. *Caltech 101* [59] was one of the first datasets hand-curated for image classification, with 101 object categories and 15-30 examples per category. One of the biggest criticisms of Caltech 101 was the lack of variability in its examples. *Caltech 256* [76] increased the number of categories to 256, while also addressing some of the shortcomings of Caltech 101. However, it still had only a handful of examples per category, and most of its images contained only a single object. *LabelMe* [201] introduced a dataset with multiple objects per category. They also provided a web interface that experts and novices could use to annotate additional images. This web interface enabled images to be labeled with polygons, helping create datasets for image segmentation.

The *Lotus Hill dataset* [271] contains a hierarchical decomposition of objects (vehicles, man-made objects, animals, etc.) along with segmentations. Only a small part of this dataset is freely available. *SUN* [261], just like LabelMe [201] and Lotus Hill [271], was curated for object detection. Pushing the size of datasets even further, *80 Million Tiny Images* [238] created a significantly larger dataset than its predecessors. It contains tiny (i.e. 32 × 32 pixels) images that were collected using WordNet [160] synsets as queries. However, because the data in 80 Million Images were not human-verified, they contain numerous errors. *YFCC100M* [234] is another large database of 100 million images that is still largely unexplored. It contains human generated and machine generated tags.

*Pascal VOC* [54] pushed research from classification to object detection with a dataset containing 20 semantic categories in 11,000 images. *ImageNet* [43] took WordNet synsets and crowdsourced a large dataset of 14 million images. They started the ILSVRC [198] challenge for a variety of computer vision tasks. Together, ILSVRC and PASCAL provide a test bench for object detection, image classification, object segmentation, person layout, and action classification. *MS-COCO* [140] recently released its dataset, with over 328,000 images with sentence descriptions and segmentations of 80 object categories. The previous largest dataset for image-based QA, *VQA* [2], contains 204,721 images annotated with three question answer pairs. They collected a dataset of 614,163 freeform questions with 6.1M ground truth answers (10 per question) and provided a baseline approach in answering questions using an image and a textual question as the input.

*Visual Genome* aims to bridge the gap between all these datasets, collecting not just annotations
Table 2.1: A comparison of existing datasets with Visual Genome. We show that Visual Genome has an order of magnitude more descriptions and question answers. It also has a more diverse set of object, attribute, and relationship classes. Additionally, Visual Genome contains a higher density of these annotations per image. The number of distinct categories in Visual Genome are calculated by lower-casing and stemming names of objects, attributes and relationships.
CHAPTER 2. VISUAL GENOME

for a large number of objects but also scene graphs, region descriptions, and question answer pairs for image regions. Unlike previous datasets, which were collected for a single task like image classification, the Visual Genome dataset was collected to be a general-purpose representation of the visual world, without bias toward a particular task. Our images contain an average of 35 objects, which is almost an order of magnitude more dense than any existing vision dataset. Similarly, we contain an average of 26 attributes and 21 relationships per image. We also have an order of magnitude more unique objects, attributes, and relationships than any other dataset. Finally, we have 1.7 million question answer pairs, also larger than any other dataset for visual question answering.

2.2.2 Image Descriptions

One of the core contributions of Visual Genome is its descriptions for multiple regions in an image. As such, we mention other image description datasets and models in this subsection. Most work related to describing images can be divided into two categories: retrieval of human-generated captions and generation of novel captions. Methods in the first category use similarity metrics between image features from predefined models to retrieve similar sentences [170, 90]. Other methods map both sentences and their images to a common vector space [170] or map them to a space of triples [56]. Among those in the second category, a common theme has been to use recurrent neural networks to produce novel captions [111, 150, 109, 247, 31, 49, 55]. More recently, researchers have also used a visual attention model [264].

One drawback of these approaches is their attention to describing only the most salient aspect of the image. This problem is amplified by datasets like Flickr 30K [275] and MS-COCO [140], whose sentence descriptions tend to focus, somewhat redundantly, on these salient parts. For example, “an elephant is seen wandering around on a sunny day,” “a large elephant in a tall grass field,” and “a very large elephant standing alone in some brush” are 3 descriptions from the MS-COCO dataset, and all of them focus on the salient elephant in the image and ignore the other regions in the image. Many real-world scenes are complex, with multiple objects and interactions that are best described using multiple descriptions [109, 135]. Our dataset pushes toward a more complete understanding of an image by collecting a dataset in which we capture not just scene-level descriptions but also myriad of low-level descriptions, the “grammar” of the scene.

2.2.3 Objects

Object detection is a fundamental task in computer vision, with applications ranging from identification of faces in photo software to identification of other cars by self-driving cars on the road. It involves classifying an object into a distinct category and localizing the object in the image. Visual Genome uses objects as a core component on which each visual scene is built. Early datasets include the face detection [92] and pedestrian datasets [48]. The PASCAL VOC and ILSVRC’s detection dataset pushed research in object detection. But the images in these datasets are iconic
and do not capture the settings in which these objects usually co-occur. To remedy this problem, MS-COCO \[140\] annotated real-world scenes that capture object contexts. However, MS-COCO was unable to describe all the objects in its images, since they annotated only 80 object categories. In the real world, there are many more objects that the ones captured by existing datasets. Visual Genome aims at collecting annotations for all visual elements that occur in images, increasing the number of distinct categories to 33,877.

2.2.4 Attributes

The inclusion of attributes allows us to describe, compare, and more easily categorize objects. Even if we haven’t seen an object before, attributes allow us to infer something about it; for example, “yellow and brown spotted with long neck” likely refers to a giraffe. Initial work in this area involved finding objects with similar features \[148\] using examplar SVMs. Next, textures were used to study objects \[241\], while other methods learned to predict colors \[61\]. Finally, the study of attributes was explicitly demonstrated to lead to improvements in object classification \[57\]. Attributes were defined to be parts (e.g. “has legs”), shapes (e.g. “spherical”), or materials (e.g. “furry”) and could be used to classify new categories of objects. Attributes have also played a large role in improving fine-grained recognition \[73\] on fine-grained attribute datasets like CUB-2011 \[250\]. In Visual Genome, we use a generalized formulation \[102\], but we extend it such that attributes are not image-specific binaries but rather object-specific for each object in a real-world scene. We also extend the types of attributes to include size (e.g. “small”), pose (e.g. “bent”), state (e.g. “transparent”), emotion (e.g. “happy”), and many more.

2.2.5 Relationships

Relationship extraction has been a traditional problem in information extraction and in natural language processing. Syntactic features \[281, 77\], dependency tree methods \[38, 20\], and deep neural networks \[223, 278\] have been employed to extract relationships between two entities in a sentence. However, in computer vision, very little work has gone into learning or predicting relationships. Instead, relationships have been implicitly used to improve other vision tasks. Relative layouts between objects have improved scene categorization \[98\], and 3D spatial geometry between objects has helped object detection \[36\]. Comparative adjectives and prepositions between pairs of objects have been used to model visual relationships and improved object localization \[78\].

Relationships have already shown their utility in improving visual cognitive tasks \[3, 269\]. A meaning space of relationships has improved the mapping of images to sentences \[56\]. Relationships in a structured representation with objects have been defined as a graph structure called a scene graph, where the nodes are objects with attributes and edges are relationships between objects. This representation can be used to generate indoor images from sentences and also to improve image search \[28, 102\]. We use a similar scene graph representation of an image that generalizes across all
these previous works [102]. Recently, relationships have come into focus again in the form of question answering about associations between objects [203]. These questions ask if a relationship, involving generally two objects, is true, e.g. “do dogs eat ice cream?”. We believe that relationships will be necessary for higher-level cognitive tasks [102, 144], so we collect the largest corpus of them in an attempt to improve tasks by actually understanding interactions between objects.

2.2.6 Question Answering

Visual question answering (QA) has been recently proposed as a proxy task of evaluating a computer vision system’s ability to understand an image beyond object recognition and image captioning [68, 146]. Several visual QA benchmarks have been proposed in the last few months. The DAQUAR [146] dataset was the first toy-sized QA benchmark built upon indoor scene RGB-D images of NYU Depth v2 [163]. Most new datasets [277, 189, 2, 66] have collected QA pairs on MS-COCO images, either generated automatically by NLP tools [189] or written by human workers [277, 2, 66].

In previous datasets, most questions concentrated on simple recognition-based questions about the salient objects, and answers were often extremely short. For instance, 90% of DAQUAR answers [146] and 89% of VQA answers [2] consist of single-word object names, attributes, and quantities. This limitation bounds their diversity and fails to capture the long-tail details of the images. Given the availability of new datasets, an array of visual QA models have been proposed to tackle QA tasks. The proposed models range from SVM classifiers and probabilistic inference [146] to recurrent neural networks [66, 147, 189] and convolutional networks [145]. Visual Genome aims to capture the details of the images with diverse question types and long answers. These questions should cover a wide range of visual tasks from basic perception to complex reasoning. Our QA dataset of 1.7 million QAs is also larger than any currently existing dataset.

2.2.7 Knowledge Representation

A knowledge representation of the visual world is capable of tackling an array of vision tasks, from action recognition to general question answering. However, it is difficult to answer “what is the minimal viable set of knowledge needed to understand about the physical world?” [82]. It was later proposed that there be a certain plurality to concepts and their related axioms [83]. These efforts have grown to model physical processes [64] or to model a series of actions as scripts [206] for stories—both of which are not depicted in a single static image but which play roles in an image’s story [243]. More recently, NELL [9] learns probabilistic horn clauses by extracting information from the web. DeepQA [62] proposes a probabilistic question answering architecture involving over 100 different techniques. Others have used Markov logic networks [282, 107] as their representation to perform statistical inference for knowledge base construction. Our work is most similar to that of those [32, 283, 284, 203] who attempt to learn common-sense relationships from images. Visual
CHAPTER 2. VISUAL GENOME

2.3 Visual Genome Data Representation

The Visual Genome dataset consists of seven main components: region descriptions, objects, attributes, relationships, region graphs, scene graphs, and question answer pairs. Figure 2.4 shows examples of each component for one image. To enable research on comprehensive understanding of images, we begin by collecting descriptions and question answers. These are raw texts without any restrictions on length or vocabulary. Next, we extract objects, attributes and relationships from our descriptions. Together, objects, attributes and relationships comprise our scene graphs that represent a formal representation of an image. In this section, we break down Figure 2.4 and explain each of the seven components. In Section 3, we will describe in more detail how data from each component is collected through a crowdsourcing platform.

2.3.1 Multiple regions and their descriptions

In a real-world image, one simple summary sentence is often insufficient to describe all the contents of and interactions in an image. Instead, one natural way to extend this might be a collection of descriptions based on different regions of a scene. In Visual Genome, we collect diverse human-generated image region descriptions, with each region localized by a bounding box. In Figure 2.5, we show three examples of region descriptions. Regions are allowed to have a high degree of overlap.
Figure 2.6: From all of the region descriptions, we extract all objects mentioned. For example, from the region description “man jumping over a fire hydrant,” we extract man and fire hydrant.

with each other when the descriptions differ. For example, “yellow fire hydrant” and “woman in shorts is standing behind the man” have very little overlap, while “man jumping over fire hydrant” has a very high overlap with the other two regions. Our dataset contains on average a total of 50 region descriptions per image. Each description is a phrase ranging from 1 to 16 words in length describing that region.

2.3.2 Multiple objects and their bounding boxes

Each image in our dataset consists of an average of 35 objects, each delineated by a tight bounding box (Figure 2.6). Furthermore, each object is canonicalized to a synset ID in WordNet [160]. For example, man would get mapped to man.n.03 (the generic use of the word to refer to any human being). Similarly, person gets mapped to person.n.01 (a human being). Afterwards, these two concepts can be joined to person.n.01 since this is a hypernym of man.n.03. We did not standardize synsets in our dataset. However, given our canonicalization, this is easily possible leveraging the WordNet ontology to avoid multiple names for one object (e.g. man, person, human), and to connect information across images.

2.3.3 A set of attributes

Each image in Visual Genome has an average of 26 attributes. Objects can have zero or more attributes associated with them. Attributes can be color (e.g. yellow), states (e.g. standing), etc. (Figure 2.7). Just like we collect objects from region descriptions, we also collect the attributes attached to these objects. In Figure 2.7, from the phrase “yellow fire hydrant,” we extract the attribute yellow for the fire hydrant. As with objects, we canonicalize all attributes to WordNet [160]: for example,
Figure 2.7: Some descriptions also provide attributes for objects. For example, the region description “yellow fire hydrant” adds that the fire hydrant is yellow. Here we show two attributes: yellow and standing.

yellow is mapped to yellow.s.01 (of the color intermediate between green and orange in the color spectrum; of something resembling the color of an egg yolk).

2.3.4 A set of relationships

Relationships connect two objects together. These relationships can be actions (e.g. jumping over), spatial (e.g. is behind), descriptive verbs (e.g. wear), prepositions (e.g. with), comparative (e.g. taller than), or prepositional phrases (e.g. drive on). For example, from the region description “man jumping over fire hydrant,” we extract the relationship jumping over between the objects man and fire hydrant (Figure 2.8). These relationships are directed from one object, called the subject, to another, called the object. In this case, the subject is the man, who is performing the relationship jumping over on the object fire hydrant. Each relationship is canonicalized to a WordNet synset ID; i.e. jumping is canonicalized to jump.a.1 (move forward by leaps and bounds). On average, each image in our dataset contains 21 relationships.

2.3.5 A set of region graphs

Combining the objects, attributes, and relationships extracted from region descriptions, we create a directed graph representation for each of the regions. Examples of region graphs are shown in Figure 2.4. Each region graph is a structured representation of a part of the image. The nodes in the graph represent objects, attributes, and relationships. Objects are linked to their respective attributes while relationships link one object to another. The links connecting two objects in Figure 2.4 point
CHAPTER 2. VISUAL GENOME

2.3.6 One scene graph

While region graphs are localized representations of an image, we also combine them into a single scene graph representing the entire image (Figure 2.3). The scene graph is the union of all region graphs and contains all objects, attributes, and relationships from each region description. By doing so, we are able to combine multiple levels of scene information in a more coherent way. For example in Figure 2.4, the leftmost region description tells us that the “fire hydrant is yellow,” while the middle region description tells us that the “man is jumping over the fire hydrant.” Together, the two descriptions tell us that the “man is jumping over a yellow fire hydrant.”

2.3.7 A set of question answer pairs

We have two types of QA pairs associated with each image in our dataset: freeform QAs, based on the entire image, and region-based QAs, based on selected regions of the image. We collect 6 different types of questions per image: what, where, how, when, who, and why. In Figure 2.4 “Q. What is the woman standing next to?; A. Her belongings” is a freeform QA. Each image has at least one question of each type listed above. Region-based QAs are collected by prompting workers with region descriptions. For example, we use the region “yellow fire hydrant” to collect the region-based QA: “Q. What color is the fire hydrant?; A. Yellow.” Region based QAs are based on the description and allow us to independently study how well models perform at answering questions using the image or the region description as input.
2.4 Dataset Statistics and Analysis

In this section, we provide statistical insights and analysis for each component of Visual Genome. Specifically, we examine the distribution of images (Section 2.4.1) and the collected data for region descriptions (Section 2.4.2) and questions and answers (Section 2.4.7). We analyze region graphs and scene graphs together in one section (Section 2.4.6), but we also break up these graph structures into their three constituent parts—objects (Section 2.4.3), attributes (Section 2.4.4), and relationships (Section 2.4.5)—and study each part individually. Finally, we describe our canonicalization pipeline and results (Section 2.4.8).
CHAPTER 2. VISUAL GENOME

2.4.1 Image Selection

The Visual Genome dataset consists of all 108,077 creative commons images from the intersection of MS-COCO’s 328,000 images and YFCC100M's 100 million images. This allows Visual Genome annotations to be utilized together with the YFCC tags and MS-COCO’s segmentations and full image captions. These images are real-world, non-iconic images that were uploaded onto Flickr by users. The images range from as small as 72 pixels wide to as large as 1280 pixels wide, with an average width of 500 pixels. We collected the WordNet synsets into which our 108,077 images can be categorized using the same method as ImageNet. Visual Genome images can be categorized into 972 ImageNet synsets. Note that objects, attributes and relationships are categorized separately into more than 18K WordNet synsets (Section 2.4.8). Figure 2.9 shows the top synsets to which our images belong. “ski” is the most common synset, with 2612 images; it is followed by “ballplayer” and “racket,” with all three synsets referring to images of people playing sports. Our dataset is somewhat biased towards images of people, as Figure 2.9 shows; however, they are quite diverse overall, as the top 25 synsets each have over 800 images, while the top 50 synsets each have over 500 examples.
CHAPTER 2. VISUAL GENOME

2.4.2 Region Description Statistics

One of the primary components of Visual Genome is its region descriptions. Every image includes an average of 50 regions with a bounding box and a descriptive phrase. Figure 2.10 shows an example image from our dataset with its 50 region descriptions. We display bounding boxes for only 6 out of the 50 descriptions in the figure to avoid clutter. These descriptions tend to be highly diverse and can focus on a single object, like in “A bag,” or on multiple objects, like in “Man taking a photo of the elephants.” They encompass the most salient parts of the image, as in “An elephant taking food from a woman,” while also capturing the background, as in “Small buildings surrounded by trees.”

MS-COCO [140] dataset is good at generating variations on a single scene-level descriptor. Consider three sentences from MS-COCO dataset on a similar image: “there is a person petting a very large elephant,” “a person touching an elephant in front of a wall,” and “a man in white
CHAPTER 2. VISUAL GENOME

Figure 2.13: The process used to convert a region description into a 300-dimensional vectorized representation.

shirt petting the cheek of an elephant.” These three sentences are single scene-level descriptions. In comparison, Visual Genome descriptions emphasize different regions in the image and thus are less semantically similar. To ensure diversity in the descriptions, we use BLEU score thresholds between new descriptions and all previously written descriptions. More information about crowdsourcing can be found in Section 3.

Region descriptions must be specific enough in an image to describe individual objects (e.g. “A bag”), but they must also be general enough to describe high-level concepts in an image (e.g. “A man being chased by a bear”). Qualitatively, we note that regions that cover large portions of the image tend to be general descriptions of an image, while regions that cover only a small fraction of the image tend to be more specific. In Figure 2.11 (a), we show the distribution of regions over the width of the region normalized by the width of the image. We see that the majority of our regions tend to be around 10% to 15% of the image width. We also note that there are a large number of regions covering 100% of the image width. These regions usually include elements like “sky,” “ocean,” “snow,” “mountains,” etc. that cannot be bounded and thus span the entire image width. In Figure 2.11 (b), we show a similar distribution over the normalized height of the region. We see a similar overall pattern, as most of our regions tend to be very specific descriptions of about 10% to 15% of the image height. Unlike the distribution over width, however, we do not see a increase in the number of regions that span the entire height of the image, as there are no common visual equivalents that span images vertically. Out of all the descriptions gathered, only one or two of them tend to be global scene descriptions that are similar to MS-COCO [140].

In Figure 2.12 we show the distribution of the length (word count) of these region descriptions. The average word count for a description is 5 words, with a minimum of 1 and a maximum of 12 words. In Figure 2.14 (a), we plot the most common phrases occurring in our region descriptions, with common stop words removed. Common visual elements like “green grass,” “tree [in] distance,” and “blue sky” occur much more often than other, more nuanced elements like “fresh strawberry.” We also study descriptions with finer precision in Figure 2.14 (b), where we plot the most common
Figure 2.14: (a) A plot of the most common visual concepts or phrases that occur in region descriptions. The most common phrases refer to universal visual concepts like “blue sky,” “green grass,” etc. (b) A plot of the most frequently used words in region descriptions. Each word is treated as an individual token regardless of which region description it came from. Colors occur the most frequently, followed by common objects like man and dog and universal visual concepts like “sky.”

Words used in descriptions. Again, we eliminate stop words from our study. Colors like “white” and “black” are the most frequently used words to describe visual concepts; we conduct a similar study on other captioning datasets including MS-COCO [140] and Flickr 30K [275] and find a similar distribution with colors occurring most frequently. Besides colors, we also see frequent occurrences of common objects like “man” and “tree” and of universal visual elements like “sky.”
Figure 2.15: (a) Example illustration showing four clusters of region descriptions and their overall themes. Other clusters not shown due to limited space. (b) Distribution of images over number of clusters represented in each image’s region descriptions. (c) We take Visual Genome with 5 random descriptions taken from each image and MS-COCO dataset with all 5 sentence descriptions per image and compare how many clusters are represented in the descriptions. We show that Visual Genome’s descriptions are more varied for a given image, with an average of 4 clusters per image, while MS-COCO’s images have an average of 2 clusters per image.
CHAPTER 2. VISUAL GENOME

<table>
<thead>
<tr>
<th></th>
<th>Visual Genome</th>
<th>ILSVRC Det. 198</th>
<th>MS-COCO Det. 140</th>
<th>Caltech101 Det. 59</th>
<th>Caltech256 Det. 76</th>
<th>PASCAL Det. 54</th>
</tr>
</thead>
<tbody>
<tr>
<td>Images</td>
<td>108,077</td>
<td>476,688</td>
<td>328,000</td>
<td>9,144</td>
<td>30,608</td>
<td>11,530</td>
</tr>
<tr>
<td>Total Objects</td>
<td>3,843,636</td>
<td>534,309</td>
<td>2,500,000</td>
<td>9,144</td>
<td>30,608</td>
<td>27,450</td>
</tr>
<tr>
<td>Total Categories</td>
<td>33,877</td>
<td>200</td>
<td>80</td>
<td>102</td>
<td>257</td>
<td>20</td>
</tr>
<tr>
<td>Objects / Category</td>
<td>113.45</td>
<td>2671.50</td>
<td>27472.50</td>
<td>90</td>
<td>119</td>
<td>1372.50</td>
</tr>
</tbody>
</table>

Table 2.2: Comparison of Visual Genome objects and categories to related datasets.

**Semantic diversity.** We also study the actual semantic contents of the descriptions. We use an unsupervised approach to analyze the semantics of these descriptions. Specifically, we use word2vec’s pre-trained model on Google news corpus to convert each word in a description to a 300-dimensional vector. Next, we remove stop words and average the remaining words to get a vector representation of the whole region description. This pipeline is outlined in Figure 2.13. We use hierarchical agglomerative clustering on vector representations of each region description and find 71 semantic and syntactic groupings or “clusters.” Figure 2.15 (a) shows four such example clusters. One cluster contains all descriptions related to tennis, like “A man swings the racquet” and “White lines on the ground of the tennis court,” while another cluster contains descriptions related to numbers, like “Three dogs on the street” and “Two people inside the tent.” To quantitatively measure the diversity of Visual Genome’s region descriptions, we calculate the number of clusters represented in a single image’s region descriptions. We show the distribution of the variety of descriptions for an image in Figure 2.15 (b). We find that on average, each image contains descriptions from 17 different clusters. The image with the least diverse descriptions contains descriptions from 4 clusters, while the image with the most diverse descriptions contains descriptions from 26 clusters.

Finally, we also compare the descriptions in Visual Genome to the captions in MS-COCO. First we aggregate all Visual Genome and MS-COCO descriptions and remove all stop words. After removing stop words, the descriptions from both datasets are roughly the same length. We conduct a similar study, in which we vectorize the descriptions for each image and calculate each dataset’s cluster diversity per image. We find that on average, 2 clusters are represented in the captions for each image in MS-COCO, with very few images in which 5 clusters are represented. Because each image in MS-COCO only contains 5 captions, it is not a fair comparison to compare the number of clusters represented in all the region descriptions in the Visual Genome dataset. We thus randomly sample 5 Visual Genome region descriptions per image and calculate the number of clusters in an image. We find that Visual Genome descriptions come from 4 or 5 clusters. We show our comparison results in Figure 2.15 (c). The difference between the semantic diversity between the two datasets is statistically significant ($t = -240, p < 0.01$).
2.4.3 Object Statistics

In comparison to related datasets, Visual Genome fares well in terms of object density and diversity (Table 2.2). Visual Genome contains approximately 35 objects per image, exceeding ImageNet [43], PASCAL [54], MS-COCO [140], and other datasets by large margins. As shown in Figure 2.17, there are more object categories represented in Visual Genome than in any other dataset. This comparison is especially pertinent with regards to Microsoft MS-COCO [140], which uses the same images as Visual Genome. The lower count of objects per category is a result of our higher number of categories. For a fairer comparison with ILSVRC 2014 Detection [198], Visual Genome has about 2239 objects per category when only the top 200 categories are considered, which is comparable to ILSVRC’s 2671.5 objects per category. For a fairer comparison with MS-COCO, Visual Genome has about 3768 objects per category when only the top 80 categories are considered. This is comparable to MS-COCO’s 140 object distribution.

The 3,843,636 objects in Visual Genome come from a variety of categories. As shown in Figure 2.18 (b), objects related to WordNet categories such as humans, animals, sports, and scenery are most common; this is consistent with the general bias in image subject matter in our dataset. Common objects like man, person, and woman occur especially frequently with occurrences of 24K, 17K, and 11K. Other objects that also occur in MS-COCO [140] are also well represented with around 5000 instances on average. Figure 2.18 (a) shows some examples of objects in images. Objects in Visual Genome span a diverse set of Wordnet categories like food, animals, and man-made structures.

It is important to look not only at what types of objects we have but also at the distribution of objects in images and regions. Figure 2.16 (a) shows, as expected, that we have between 0 and 2 objects in each region on average. It is possible for regions to contain no objects if their descriptions refer to no explicit objects in the image. For example, a region described as “it is dark outside”
has no objects to extract. Regions with only one object generally have descriptions that focus on the attributes of a single object. On the other hand, regions with two or more objects generally have descriptions that contain both attributes of specific objects and relationships between pairs of objects.

As shown in Figure 2.16 (b), each image contains on average around 35 distinct objects. Few images have an extremely high number of objects (e.g. over 40). Due to the image biases that exist in the dataset, we have twice as many annotations for men than we do of women.

2.4.4 Attribute Statistics

Attributes allow for detailed description and disambiguation of objects in our dataset. Our dataset contains 2.8 million total attributes with 68,111 unique attributes. Attributes include colors (e.g. green), sizes (e.g. tall), continuous action verbs (e.g. standing), materials (e.g. plastic), etc. Each object can have multiple attributes.

On average, each image in Visual Genome contains 26 attributes (Figure 2.19). Each region contains on average 1 attribute, though about 34% of regions contain no attribute at all; this is primarily because many regions are relationship-focused. Figure 2.20 (a) shows the distribution of the most common attributes in our dataset. Colors (e.g. white, green) are by far the most frequent
CHAPTER 2. VISUAL GENOME

Figure 2.18: (a) Examples of objects in Visual Genome. Each object is localized in its image with a tightly drawn bounding box. (b) Plot of the most frequently occurring objects in images. People are the most frequently occurring objects in our dataset, followed by common objects and visual elements like building, shirt, and sky.

attributes. Also common are sizes (e.g. large) and materials (e.g. wooden). Figure 2.20 (b) shows the distribution of attributes describing people (e.g. man, girls, and person). The most common attributes describing people are intransitive verbs describing their states of motion (e.g. standing and walking). Certain sports (e.g. skiing, surfboarding) are overrepresented due to an image bias towards these sports.
Attribute Graphs. We also qualitatively analyze the attributes in our dataset by constructing co-occurrence graphs, in which nodes are unique attributes and edges connect those attributes that describe the same object. For example, if an image contained a “large black dog” (large(dog), black(dog)) and another image contained a “large yellow cat” (large(cat), yellow(cat)), its attributes would form an incomplete graph with edges (large, black) and (large, yellow). We create two such graphs: one for the total set of attributes and a second where we consider only objects that refer to people. A subgraph of the 16 most frequently connected (co-occurring) person-related attributes is shown in Figure 2.21 (a).

Cliques in these graphs represent groups of attributes in which at least one co-occurrence exists for each pair of attributes. In the previous example, if a third image contained a “black and yellow taxi” (black(taxi), yellow(taxi)), the resulting third edge would create a clique between the attributes black, large, and yellow. When calculated across the entire Visual Genome dataset, these cliques provide insight into commonly perceived traits of different types of objects. Figure 2.21 (b) is a selected representation of three example cliques and their overlaps. From just a clique of attributes, we can predict what types of objects are usually referenced. In Figure 2.21 (b), we see that these cliques describe an animal (left), water body (top right), and human hair (bottom right).

Other cliques (not shown) can also uniquely identify object categories. In our set, one clique
Figure 2.20: (a) Distribution showing the most common attributes in the dataset. Colors (e.g. white, red) and materials (e.g. wooden, metal) are the most common. (b) Distribution showing the number of attributes describing people. State-of-motion verbs (e.g. standing, walking) are the most common, while certain sports (e.g. skiing, surfing) are also highly represented due to an image source bias in our image set.

contains athletic, young, fit, skateboarding, focused, teenager, male, skinny, and happy, capturing some of the common traits of skateboarders in our set. Another such clique has shiny, small, metal, silver, rusty, parked, and empty, most likely describing a subset of cars. From these cliques, we can thus infer distinct objects and object types based solely on their attributes, potentially allowing for highly specific object identification based on selected characteristics.
2.4.5 Relationship Statistics

Relationships are the core components that link objects in our scene graphs. Relationships are directional, i.e. they involve two objects, one acting as the subject and one as the object of a
CHAPTER 2. VISUAL GENOME

Figure 2.22: Distribution of relationships (a) per image region, (b) per image object, (c) per image.

Figure 2.22: Distribution of relationships (a) per image region, (b) per image object, (c) per image.

Figure 2.22: Distribution of relationships (a) per image region, (b) per image object, (c) per image.

Figure 2.22: Distribution of relationships (a) per image region, (b) per image object, (c) per image.

Figure 2.22: Distribution of relationships (a) per image region, (b) per image object, (c) per image.

Figure 2.22: Distribution of relationships (a) per image region, (b) per image object, (c) per image.

Figure 2.22: Distribution of relationships (a) per image region, (b) per image object, (c) per image.

Figure 2.22: Distribution of relationships (a) per image region, (b) per image object, (c) per image.

Figure 2.22: Distribution of relationships (a) per image region, (b) per image object, (c) per image.

Top relationship distributions. We display the most frequently occurring relationships in Figure 2.23 (a). On is the most common relationship in our dataset. This is primarily because of the flexibility of the word on, which can refer to spatial configuration (on top of), attachment (hanging on), etc. Other common relationships involve actions like holding and wearing and spatial configurations like behind, next to, and under. Figure 2.23 (b) shows a similar distribution but for relationships involving people. Here we notice more human-centric relationships or actions such as kissing, chatting with, and talking to. The two distributions follow a
Table 2.3: The average number of objects, attributes, and relationships per region graph and per scene graph.

<table>
<thead>
<tr>
<th></th>
<th>Objects</th>
<th>Attributes</th>
<th>Relationships</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region Graph</td>
<td>0.71</td>
<td>0.52</td>
<td>0.43</td>
</tr>
<tr>
<td>Scene Graph</td>
<td>35</td>
<td>26</td>
<td>21</td>
</tr>
</tbody>
</table>

Zipf distribution.

**Understanding affordinges.** Relationships allow us to also understand the affordances of objects. Figure 2.24(a) shows the distribution for subjects while Figure 2.24(b) shows a similar distribution for objects. Comparing the two, we find clear patterns of people-like subject entities such as person, man, policeman, boy, and skateboarder that can ride other objects; the other distribution contains objects that afford riding, such as horse, bike, elephant, motorcycle, and skateboard. We can also learn specific common-sense knowledge, like that zebras eat hay and grass while a person eats pizzas and burgers and that couches usually have pillows on them.

**Related work comparison.** It is also worth mentioning in this section some prior work on relationships. The concept of visual relationships has already been explored in Visual Phrases [204], who introduced a dataset of 17 such relationships such as next_to(person, bike) and riding(person, horse). However, their dataset is limited to just these 17 relationships. Similarly, the MS-COCO-a a scene graph dataset [195] introduced 156 actions that humans performed in MS-COCO’s dataset [140]. They show that to exhaustively describe “common” images involving humans, only a small set of visual actions is needed. However, their dataset is limited to just actions, while our relationships are more general and numerous, with over 42,374 unique relationships. Finally, VisKE [203] introduced 6500 relationships, but in a much smaller dataset of images than Visual Genome.

### 2.4.6 Region and Scene Graph Statistics

We introduce in this paper the largest dataset of scene graphs to date. We use these graph representations of images as a deeper understanding of the visual world. In this section, we analyze the properties of these representations, both at the region-level through region graphs and at the image level through scene graphs. We also briefly explore other datasets with scene graphs and provide aggregate statistics on our entire dataset.

In previous work, scene graphs have been collected by asking humans to write a list of triples about an image [102]. However, unlike them, we collect graphs at a much more fine-grained level: the region graph. We obtained our graphs by asking workers to create them from the descriptions we collected from our regions. Therefore, we end up with multiple graphs for an image, one for every region description. Together, we can combine all the individual region graphs to aggregate a scene
Figure 2.23: (a) A sample of the most frequent relationships in our dataset. In general, the most common relationships are spatial (on top of, on side of, etc.). (b) A sample of the most frequent relationships involving humans in our dataset. The relationships involving people tend to be more action oriented (walk, speak, run, etc.).

graph for an image. This scene graph is made up of all the individual region graphs. In our scene graph representation, we merge all the objects that referenced by multiple region graphs into one node in the scene graph.

Each of our images has between 5 to 100 region graphs per image, with an average of 50. Each image has exactly one scene graph. Note that the number of region descriptions and the number
Figure 2.24: (a) Distribution of subjects for the relationship riding. (b) Distribution of objects for the relationship riding. Subjects comprise of people-like entities like person, man, policeman, boy, and skateboarder that can ride other objects. On the other hand, objects like horse, bike, elephant, and motorcycle are entities that can afford riding.

of region graphs for an image are not the same. For example, consider the description “it is a sunny day”. Such a description contains no objects, which are the building blocks of a region graph. Therefore, such descriptions have no region graphs associated with them.

Objects, attributes, and relationships occur as a normal distribution in our data. Table 2.3 shows that in a region graph, there are an average of 0.71 objects, 0.52 attributes, and 21 relationships. Each scene graph and consequently each image has average of 35 objects, 26 attributes, and 21
2.4.7 Question Answering Statistics

We collected 1,773,258 question answering (QA) pairs on the Visual Genome images. Each pair consists of a question and its correct answer regarding the content of an image. On average, every image has 17 QA pairs. Rather than collecting unconstrained QA pairs as previous work has done \[2, 60, 146\], each question in Visual Genome starts with one of the six Ws – what, where, when, who, why, and how. There are two major benefits to focusing on six types of questions. First, they offer a considerable coverage of question types, ranging from basic perceptual tasks (e.g. recognizing objects and scenes) to complex common sense reasoning (e.g. inferring motivations of people and causality of events). Second, these categories present a natural and consistent stratification of task difficulty, indicated by the baseline performance in Section 6.1.4. For instance, why questions that involve complex reasoning lead to the poorest performance (3.4% top-100 accuracy compared to 9.6% top-100 accuracy of the next lowest) of the six categories. This enables us to obtain a better understanding of the strengths and weaknesses of today’s computer vision models, which sheds light on future directions in which to proceed.

We now analyze the diversity and quality of our questions and answers. Our goal is to construct a large-scale visual question answering dataset that covers a diverse range of question types, from basic cognition tasks to complex reasoning tasks. We demonstrate the richness and diversity of our QA pairs by examining the distributions of questions and answers in Figure 2.25.

Figure 2.25: Example QA pairs in the Visual Genome dataset. Our QA pairs cover a spectrum of visual tasks from recognition to high-level reasoning.
Question type distributions. The questions naturally fall into the 6W categories via their interrogative words. Inside each of the categories, the second and following words categorize the questions with increasing granularity. Inspired by VQA [2], we show the distributions of the questions by their first three words in Figure 2.26a. We can see that “what” is the most common of the six categories. A notable difference between our question distribution and VQA’s is that we focus on ensuring that all six question categories are adequately represented, while in VQA, 38.37% of the questions are yes/no binary questions. As a result, a trivial model can achieve a reasonable performance by just predicting “yes” or “no” as answers. We encourage more difficult QA pairs by ruling out binary questions.

Question and answer length distributions. We also analyze the question and answer lengths of each 6W category. Figure 2.26b shows the average question and answer lengths of each category. Overall, the average question and answer lengths are 5.7 and 1.8 words respectively. In contrast to the VQA dataset, where 89.32%, 6.91%, and 2.74% of the answers consist of one, two, or three words, our answers exhibit a long-tail distribution where 57.3%, 18.1%, and 15.7% of the answers
have one, two, or three words respectively. We avoid verbosity by instructing the workers to write answers as concisely as possible. The coverage of long answers means that many answers contain a short description that contains more details than merely an object or an attribute. It shows the richness and complexity of our visual QA tasks beyond object-centric recognition tasks. We foresee that these long-tail answers can motivate future research in common-sense reasoning and high-level image understanding.

Figure 2.27: An example image from the Visual Genome dataset with its region descriptions, QA pairs, objects, attributes, and relationships canonicalized. The large text boxes are WordNet synsets referenced by this image. For example, the carriage is mapped to carriage.n.02: a vehicle with wheels drawn by one or more horses. We do not show the bounding boxes for the objects in order to allow readers to see the image clearly. We also only show a subset of the scene graph for this image to avoid cluttering the figure.

### 2.4.8 Canonicalization Statistics

In order to reduce the ambiguity in the concepts of our dataset and connect it to other resources used by the research community, we canonicalize the semantic meanings of all objects, relationships, and attributes in Visual Genome. By “canonicalization,” we refer to word sense disambiguation (WSD) by mapping the components in our dataset to their respective synsets in the WordNet ontology. This mapping reduces the noise in the concepts contained in the dataset and also facilitates the linkage between Visual Genome and other data sources such as ImageNet, which is built on top of the WordNet ontology.

Figure 2.27 shows an example image from the Visual Genome dataset with its components canonicalized. For example, horse is canonicalized as horse.n.01: solid-hoofed herbivorous quadruped domesticated since prehistoric times. Its attribute, clydesdale, is canonicalized as its breed clydesdale.n.01: heavy feathered-legged breed of draft horse originally from Scotland. We also show an example of a QA from which we extract the
Table 2.4: Precision, recall, and mapping accuracy percentages for object, attribute, and relationship canonicalization.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objects</td>
<td>88.0</td>
<td>98.5</td>
</tr>
<tr>
<td>Attributes</td>
<td>85.7</td>
<td>95.9</td>
</tr>
<tr>
<td>Relationships</td>
<td>92.9</td>
<td>88.5</td>
</tr>
</tbody>
</table>

nouns shamrocks, symbol, and St. Patrick’s day, all of which we canonicalize to WordNet as well.

**Related work.** Canonicalization, or WSD \[172\], has been used in numerous applications, including machine translation, information retrieval, and information extraction \[197, 134\]. In English sentences, sentences like “He scored a goal” and “It was his goal in life” carry different meanings for the word “goal.” Understanding these differences is crucial for translating languages and for returning correct results for a query. Similarly, in Visual Genome, we ensure that all our components are canonicalized to understand how different objects are related to each other; for example, “person” is a hypernym of “man” and “woman.” Most past canonicalization models use precision, recall, and F1 score to evaluate on the Semeval dataset \[157\]. The current state-of-the-art performance on Semeval is an F1 score of 75\% \[33\]. Since our canonicalization setup is different from the Semeval benchmark (we have an open vocabulary and no annotated ground truth for evaluation), our canonicalization method is not directly comparable to these existing methods. We do however, achieve a similar precision and recall score on a held-out test set described below.

**Region descriptions and QAs.** We canonicalize all objects mentioned in all region descriptions and QA pairs. Because objects need to be extracted from the phrase text, we use Stanford NLP tools \[149\] to extract the noun phrases in each region description and QA, resulting in 99\% recall of noun phrases from a subset of 200 region descriptions we manually annotated. After obtaining the noun phrases, we map each to its most frequent matching synset (according to WordNet lexeme counts). This resulted in an overall mapping accuracy of 88\% and a recall of 98.5\% (Figure 2.4). The most common synsets extracted from region descriptions, QAs, and objects are shown in Figure 2.28.

**Attributes.** We canonicalize attributes from the crowd-extracted attributes present in our scene graphs. The “attribute” designation encompasses a wide range of grammatical parts of speech. Because part-of-speech taggers rely on high-level syntax information and thus fail on the disjoint elements of our scene graphs, we normalize each attribute based on morphology alone (so-called “stemming” \[10\]). Then, as with objects, we map each attribute phrase to the most frequent matching WordNet synset. We include 15 hand-mapped rules to address common failure cases in which WordNet’s frequency counts prefer abstract senses of words over the spatial senses present.
in visual data, e.g. short.a.01: limited in duration over short.a.02: lacking in length. For verification, we randomly sample 200 attributes, produce ground-truth mappings by hand, and compare them to the results of our algorithm. This resulted in a recall of 95.9% and a mapping accuracy of 85.7%. The most common attribute synsets are shown in Figure 2.29 (a).
As with attributes, we canonicalize the relationships isolated in our scene graphs. We exclude prepositions, which are not recognized in WordNet, leaving a set primarily composed of verb relationships. Since the meanings of verbs are highly dependent upon their morphology and syntactic placement (e.g. passive cases, prepositional phrases), we map the structure of each relationship to the appropriate WordNet sentence frame and only consider those WordNet synsets with matching sentence frames. For each verb-synset pair, we then consider the root hypernym.
of that synset to reduce potential noise from WordNet’s fine-grained sense distinctions. We also include 20 hand-mapped rules, again to correct for WordNet’s lower representation of concrete or spatial senses; for example, the concrete hold.v.02: have or hold in one’s hand or grip is less frequent in WordNet than the abstract hold.v.01: cause to continue in a certain state. For verification, we again randomly sample 200 relationships and compare the results of our canonicalization against ground-truth mappings. This resulted in a recall of 88.5% and a mapping accuracy of 92.9%. While several datasets, such as VerbNet [209] and FrameNet [4], include semantic restrictions or frames to improve classification, there is no comprehensive method of mapping to those restrictions or frames. The most common relationship synsets are shown in Figure 2.29 (b).
Chapter 3

Crowdsourcing Strategies

Visual Genome was collected and verified entirely by crowd workers from Amazon Mechanical Turk. In this section, we outline the pipeline employed in creating all the components of the dataset. Each component (region descriptions, objects, attributes, relationships, region graphs, scene graphs, questions and answers) involved multiple task stages. We mention the different strategies used to make our data accurate and to enforce diversity in each component. We also provide background information about the workers who helped make Visual Genome possible.

3.0.1 Crowd Workers

We used Amazon Mechanical Turk (AMT) as our primary source of annotations. Overall, a total of over 33,000 unique workers contributed to the dataset. The dataset was collected over the course of 6 months after 15 months of experimentation and iteration on the data representation. Approximately 800,000 Human Intelligence Tasks (HITs) were launched on AMT, where each HIT involved creating descriptions, questions and answers, or region graphs. Each HIT was designed such that workers manage to earn anywhere between $6-$8 per hour if they work continuously, in line with ethical research standards on Mechanical Turk [205]. Visual Genome HITs achieved a 94.1% retention rate, meaning that 94.1% of workers who completed one of our tasks went ahead to do more. Table 3.1 outlines the percentage distribution of the locations of the workers. 93.02% of workers contributed from the United States.

Figures 3.1(a) and (b) outline the demographic distribution of our crowd workers. This data was collected using a survey HIT. The majority of our workers were between the ages of 25 and 34 years old. Our youngest contributor was 18 years and the oldest was 68 years old. We also had a near-balanced split of 54.15% male and 45.85% female workers.


### Table 3.1: Geographic distribution of countries from where crowd workers contributed to Visual Genome.

<table>
<thead>
<tr>
<th>Country</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>93.02%</td>
</tr>
<tr>
<td>Philippines</td>
<td>1.29%</td>
</tr>
<tr>
<td>Kenya</td>
<td>1.13%</td>
</tr>
<tr>
<td>India</td>
<td>0.94%</td>
</tr>
<tr>
<td>Russia</td>
<td>0.50%</td>
</tr>
<tr>
<td>Canada</td>
<td>0.47%</td>
</tr>
<tr>
<td>(Others)</td>
<td>2.65%</td>
</tr>
</tbody>
</table>

Figure 3.1: (a) Age and (b) gender distribution of Visual Genome’s crowd workers.

#### 3.0.2 Region Descriptions

Visual Genome’s main goal is to enable the study of cognitive computer vision tasks. The next step towards understanding images requires studying relationships between objects in scene graph representations of images. However, we observed that collecting scene graphs directly from an image leads to workers annotating easy, frequently-occurring relationships like `wearing(man, shirt)` instead of focusing on salient parts of the image. This is evident from previous datasets \[102, 144\] that contain a large number of such relationships. After experimentation, we observed that when asked to describe an image using natural language, crowd workers naturally start with the most salient part of the image and then move to describing other parts of the image one by one. Inspired by this finding, we focused our attention towards collecting a dataset of region descriptions that is diverse in content.

When a new image is added to the crowdsourcing pipeline with no annotations, it is sent to a worker who is asked to draw three bounding boxes and write three descriptions for the region enclosed by each box. Next, the image is sent to another worker along with the previously written descriptions. Workers are explicitly encouraged to write descriptions that have not been written before. This process is repeated until we have collected 50 region descriptions for each image. To prevent workers from having to skim through a long list of previously written descriptions, we only show them the top seven most similar descriptions. We calculate these most similar descriptions
using BLEU-like \[175\] (n-gram) scores between pairs of sentences. We define the similarity score \( S \) between a description \( d_i \) and a previous description \( d_j \) to be:

\[
S_n(d_i, d_j) = b(d_i, d_j) \exp \left( \frac{1}{N} \sum_{n=1}^{N} \log p_n(d_i, d_j) \right)
\]

(3.1)

where we enforce a brevity penalty using:

\[
b(d_i, d_j) = \begin{cases} 
1 & \text{if } \text{len}(d_i) > \text{len}(d_j) \\
\exp\left(-\frac{\text{len}(d_i)}{\text{len}(d_j)}\right) & \text{otherwise}
\end{cases}
\]

(3.2)

and \( p_n \) calculates the percentage of n-grams in \( d_i \) that match n-grams in \( d_j \).

When a worker writes a new description, we programmatically enforce that it has not been repeated by using BLEU score thresholds set to 0.7 to ensure that it is dissimilar to descriptions from both of the following two lists:

1. **Image-specific descriptions.** A list of all previously written descriptions for that image.

2. **Global image descriptions.** A list of the top 100 most common written descriptions of all images in the dataset. This prevents very common phrases like “sky is blue” from dominating the set of region descriptions. The list of top 100 global descriptions is continuously updated as more data comes in.

Finally, we ask workers to draw bounding boxes that satisfy one requirement: **coverage**. The bounding box must cover all objects mentioned in the description. Figure 3.2 shows an example of a good box that covers both the street as well as the car mentioned in the description, as well as an example of a bad box.

### 3.0.3 Objects

Once 50 region descriptions are collected for an image, we extract the visual objects from each description. Each description is sent to one crowd worker, who extracts all the objects from the description and grounds each object as a bounding box in the image. For example, from Figure 2.4, let’s consider the description “woman in shorts is standing behind the man.” A worker would extract three objects: woman, shorts, and man. They would then draw a box around each of the objects. We require each bounding box to be drawn to satisfy two requirements: **coverage** and **quality**. Coverage has the same definition as described above in Section 3.0.2, where we ask workers to make sure that the bounding box covers the object completely (Figure 3.3). Quality requires that each bounding box be as tight as possible around its object such that if the box’s length or height were
decreased by one pixel, it would no longer satisfy the coverage requirement. Since a one pixel error can be physically impossible for most workers, we relax the definition of quality to four pixels.

Multiple descriptions for an image might refer to the same object, sometimes with different words. For example, a man in one description might be referred to as person in another description. We can thus use this crowdsourcing stage to build these co-reference chains. With each region description given to a worker to process, we include a list of previously extracted objects as suggestions. This allows a worker to choose a previously drawn box annotated as man instead of redrawing a new box for person.

Finally, to increase the speed with which workers complete this task, we also use Stanford’s dependency parser \cite{149} to extract nouns automatically and send them to the workers as suggestions. While the parser manages to find most of the nouns, it sometimes misses compound nouns, so we avoided completely depending on this automated method. By combining the parser with crowdsourcing tasks, we were able to speed up our object extraction process without losing accuracy.

3.0.4 Attributes, Relationships, and Region Graphs

Once all objects have been extracted from each region description, we can extract the attributes and relationships described in the region. We present each worker with a region description along with its extracted objects and ask them to add attributes to objects or to connect pairs of objects with relationships, based on the text of the description. From the description “woman in shorts is standing behind the man”, workers will extract the attribute standing for the woman and the relationships in(\textit{woman, shorts}) and behind(\textit{woman, man}). Together, objects, attributes, and relationships form the region graph for a region description. Some descriptions like “it is a sunny day” do not contain any objects and therefore have no region graphs associated with them. Workers are asked to not
generate any graphs for such descriptions. We create scene graphs by combining all the region graphs for an image by combining all the co-referenced objects from different region graphs.

### 3.0.5 Scene Graphs

The scene graph is the union of all region graphs extracted from region descriptions. We merge nodes from region graphs that correspond to the same object; for example, man and person in two different region graphs might refer to the same object in the image. We say that objects from different graphs refer to the same object if their bounding boxes have an intersection over union of 0.9. However, this heuristic might contain false positives. So, before merging two objects, we ask workers to confirm that a pair of objects with significant overlap are indeed the same object. For example, in Figure 3.4 (right), the fox might be extracted from two different region descriptions. These boxes are then combined together (Figure 3.4 (left)) when constructing the scene graph.

### 3.0.6 Questions and Answers

To create question answer (QA) pairs, we ask the AMT workers to write pairs of questions and answers about an image. To ensure quality, we instruct the workers to follow three rules: 1) start the questions with one of the “six Ws” (who, what, where, when, why and how); 2) avoid ambiguous and speculative questions; 3) be precise and unique, and relate the question to the image such that it is clearly answerable if and only if the image is shown.

We collected two separate types of QAs: freeform QAs and region-based QAs. In freeform QA, we ask a worker to look at an image and write eight QA pairs about it. To encourage diversity, we enforce that workers write at least three different Ws out of the six in their eight pairs. In
region-based QA, we ask the workers to write a pair based on a given region. We select the regions that have large areas (more than 5k pixels) and long phrases (more than 4 words). This enables us to collect around twenty region-based pairs at the same cost of the eight freeform QAs. In general, freeform QA tends to yield more diverse QA pairs that enrich the question distribution; region-based QA tends to produce more factual QA pairs at a lower cost.

3.0.7 Verification

All Visual Genome data go through a verification stage as soon as they are annotated. This stage helps eliminate incorrectly labeled objects, attributes, and relationships. It also helps remove region descriptions and questions and answers that might be correct but are vague (“This person seems to enjoy the sun.”), subjective (“room looks dirty”), or opinionated (“Being exposed to hot sun like this may cause cancer”).

Verification is conducted using two separate strategies: majority voting [222] and rapid judgments [116]. All components of the dataset except objects are verified using majority voting. Majority voting [222] involves three unique workers looking at each annotation and voting on whether it is factually correct. An annotation is added to our dataset if at least two (a majority) out of the three workers verify that it is correct.

We only use rapid judgments to speed up the verification of the objects in our dataset. Rapid judgments [116] use an interface inspired by rapid serial visual processing that enable verification of objects with an order of magnitude increase in speed than majority voting.
3.0.8 Canonicalization

All the descriptions and QAs that we collect are freeform worker-generated texts. They are not constrained by any limitations. For example, we do not force workers to refer to a man in the image as a man. We allow them to choose to refer to the man as person, boy, man, etc. This ambiguity makes it difficult to collect all instances of man from our dataset. In order to reduce the ambiguity in the concepts of our dataset and connect it to other resources used by the research community, we map all objects, attributes, relationships, and noun phrases in region descriptions and QAs to synsets in WordNet [160]. In the example above, person, boy, and man would map to the synsets: person.n.01 (a human being), male_child.n.01 (a youthful male person) and man.n.03 (the generic use of the word to refer to any human being) respectively. Thanks to the WordNet hierarchy it is now possible to fuse those three expressions of the same concept into person.n.01 (a human being), which is the lowest common ancestor node of all aforementioned synsets.

We use the Stanford NLP tools [149] to extract the noun phrases from the region descriptions and QAs. Next, we map them to their most frequent matching synset in WordNet according to WordNet lexeme counts. We then refine this simple heuristic by hand-crafting mapping rules for the 30 most common failure cases. For example according to WordNet’s lexeme counts the most common semantic for “table” is table.n.01 (a set of data arranged in rows and columns). However in our data it is more likely to see pieces of furniture and therefore bias the mapping towards table.n.02 (a piece of furniture having a smooth flat top that is usually supported by one or more vertical legs). The objects in our scene graphs are already noun phrases and are mapped to WordNet in the same way.

We normalize each attribute based on morphology (so called “stemming”) and map them to the WordNet adjectives. We include 15 hand-crafted rules to address common failure cases, which typically occur when the concrete or spatial sense of the word seen in an image is not the most common overall sense. For example, the synset long.a.02 (of relatively great or greater than average spatial extension) is less common in WordNet than long.a.01 (indicating a relatively great or greater than average duration of time), even though instances of the word “long” in our images are much more likely to refer to that spatial sense.

For relationships, we ignore all prepositions as they are not recognized by WordNet. Since the meanings of verbs are highly dependent upon their morphology and syntactic placement (e.g. passive cases, prepositional phrases), we try to find WordNet synsets whose sentence frames match with the context of the relationship. Sentence frames in WordNet are formalized syntactic frames in which a certain sense of a word might appear; e.g., play.v.01: participate in games or sport occurs in the sentence frames “Somebody [play]s” and “Somebody [play]s something.” For each verb-synset pair, we then consider the root hypernym of that synset to reduce potential
noise from WordNet’s fine-grained sense distinctions. The WordNet hierarchy for verbs is segmented and originates from over 100 root verbs. For example, draw.v.01: cause to move by pulling traces back to the root hyponym move.v.02: cause to move or shift into a new position, while draw.v.02: get or derive traces to the root get.v.01: come into the possession of something concrete or abstract. We also include 20 hand-mapped rules, again to correct for WordNet’s lower representation of concrete or spatial senses.

These mappings are not perfect and still contain some ambiguity. Therefore, we send all our mappings along with the top four alternative synsets for each term to AMT. We ask workers to verify that our mapping was accurate and change the mapping to an alternative one if it was a better fit. We present workers with the concept we want to canonicalize along with our proposed corresponding synset with 4 additional options. To prevent workers from always defaulting to the our proposed synset, we do not explicitly specify which one of the 5 synsets presented is our proposed synset. Section 2.4.8 provides experimental precision and recall scores for our canonicalization strategy.
Chapter 4

Embracing Error to Enable Rapid Crowdsourcing

4.1 Introduction

Social science [112, 154], interactive systems [58, 125] and machine learning [43, 140] are becoming more and more reliant on large-scale, human-annotated data. Increasingly large annotated datasets have unlocked a string of social scientific insights [69, 21] and machine learning performance improvements [120, 71, 247]. One of the main enablers of this growth has been microtask crowdsourcing [222]. Microtask crowdsourcing marketplaces such as Amazon Mechanical Turk offer a scale and cost that makes such annotation feasible. As a result, companies are now using crowd work to complete hundreds of thousands of tasks per day [151].

However, even microtask crowdsourcing can be insufficiently scalable, and it remains too expensive for use in the production of many industry-size datasets [103]. Cost is bound to the amount of work completed per minute of effort, and existing techniques for speeding up labeling (reducing the amount of required effort) are not scaling as quickly as the volume of data we are now producing that must be labeled [235]. To expand the applicability of crowdsourcing, the number of items annotated per minute of effort needs to increase substantially.

In this paper, we focus on one of the most common classes of crowdsourcing tasks [94]: binary annotation. These tasks are yes-or-no questions, typically identifying whether or not an input has a specific characteristic. Examples of these types of tasks are topic categorization (e.g., “Is this article about finance?”) [207], image classification (e.g., “Is this a dog?”) [43, 140, 138], audio styles [211] and emotion detection [138] in songs (e.g., “Is the music calm and soothing?”), word similarity (e.g., “Are...

Our method for enabling rapid crowdsourcing was also a highly collaborative project, in which my main contributions were designing the rapid interface and distributing the task to collect annotations on Amazon Mechanical Turk.
CHAPTER 4. EMBRACING ERROR TO ENABLE RAPID CROWDSOURCING

Figure 4.1: (a) Images are shown to workers at 100ms per image. Workers react whenever they see a dog. (b) The true labels are the ground truth dog images. (c) The workers’ keypresses are slow and occur several images after the dog images have already passed. We record these keypresses as the observed labels. (d) Our technique models each keypress as a delayed Gaussian to predict (e) the probability of an image containing a dog from these observed labels.

*shipment* and *cargo* synonyms?*) 161 and sentiment analysis (e.g., “Is this tweet positive?”) 174.

Previous methods have sped up binary classification tasks by minimizing worker error. A central assumption behind this prior work has been that workers make errors because they are not trying hard enough (e.g., “a lack of expertise, dedication [or] interest” 214). Platforms thus punish errors harshly, for example by denying payment. Current methods calculate the minimum redundancy necessary to be confident that errors have been removed 214, 220, 221. These methods typically result in a 0.25× to 1× speedup beyond a fixed majority vote 178, 200, 214, 108.

We take the opposite position: that designing the task to encourage some error, or even make errors inevitable, can produce far greater speedups. Because platforms strongly punish errors, workers carefully examine even straightforward tasks to make sure they do not represent edge cases 153, 96. The result is slow, deliberate work. We suggest that there are cases where we can encourage workers to move quickly by telling them that making some errors is acceptable. Though individual worker accuracy decreases, we can recover from these mistakes post-hoc algorithmically (Figure 4.1).

We manifest this idea via a crowdsourcing technique in which workers label a rapidly advancing stream of inputs. Workers are given a binary question to answer, and they observe as the stream automatically advances via a method inspired by rapid serial visual presentation (RSVP) 137, 60. Workers press a key whenever the answer is “yes” for one of the stream items. Because the stream is advancing rapidly, workers miss some items and have delayed responses. However, workers are reassured that the requester expects them to miss a few items. To recover the correct answers, the technique randomizes the item order for each worker and model workers’ delays as a normal distribution whose variance depends on the stream’s speed. For example, when labeling whether
images have a “barking dog” in them, a self-paced worker on this task takes 1.7s per image on average. With our technique, workers are shown a stream at 100ms per image. The technique models the delays experienced at different input speeds and estimates the probability of intended labels from the key presses.

We evaluate our technique by comparing the total worker time necessary to achieve the same precision on an image labeling task as a standard setup with majority vote. The standard approach takes three workers an average of 1.7s each for a total of 5.1s. Our technique achieves identical precision (97%) with five workers at 100ms each, for a total of 500ms of work. The result is an order of magnitude speedup of $10^\times$.

This relative improvement is robust across both simple tasks, such as identifying dogs, and complicated tasks, such as identifying “a person riding a motorcycle” (interactions between two objects) or “people eating breakfast” (understanding relationships among many objects). We generalize our technique to other tasks such as word similarity detection, topic classification and sentiment analysis. Additionally, we extend our method to categorical classification tasks through a ranked cascade of binary classifications. Finally, we test workers’ subjective mental workload and find no measurable increase.

**Contributions.** We make the following contributions:

1. We introduce a rapid crowdsourcing technique that makes errors normal and even inevitable. We show that it can be used to effectively label large datasets by achieving a speedup of an order of magnitude on several binary labeling crowdsourcing tasks.

2. We demonstrate that our technique can be generalized to multi-label categorical labeling tasks, combined independently with existing optimization techniques, and deployed without increasing worker mental workload.

### 4.2 Related Work

The main motivation behind our work is to provide an environment where humans can make decisions quickly. We encourage a margin of human error in the interface that is then rectified by inferring the true labels algorithmically. In this section, we review prior work on crowdsourcing optimization and other methods for motivating contributions. Much of this work relies on artificial intelligence techniques: we complement this literature by changing the crowdsourcing interface rather than focusing on the underlying statistical model.

Our technique is inspired by rapid serial visual presentation (RSVP), a technique for consuming media rapidly by aligning it within the foveal region and advancing between items quickly \[137 \ 60\]. RSVP has already been proven to be effective at speeding up reading rates \[258\]. RSVP users can react to a single target image in a sequence of images even at 125ms per image with 75% accuracy \[182\]. However, when trying to recognize concepts in images, RSVP only achieves an
accuracy of 10% at the same speed \cite{183}. In our work, we integrate multiple workers’ errors to successfully extract true labels.

Many previous papers have explored ways of modeling workers to remove bias or errors from ground truth labels \cite{257, 256, 280, 178, 95}. For example, an unsupervised method for judging worker quality can be used as a prior to remove bias on binary verification labels \cite{95}. Individual workers can also be modeled as projections into an open space representing their skills in labeling a particular image \cite{257}. Workers may have unknown expertise that may in some cases prove adversarial to the task. Such adversarial workers can be detected by jointly learning the difficulty of labeling a particular datum along with the expertise of workers \cite{256}. Finally, a generative model can be used to model workers’ skills by minimizing the entropy of the distribution over their labels and the unknown true labels \cite{280}. We draw inspiration from this literature, calibrating our model using a similar generative approach to understand worker reaction times. We model each worker’s reaction as a delayed Gaussian distribution.

In an effort to reduce cost, many previous papers have studied the tradeoffs between speed (cost) and accuracy on a wide range of tasks \cite{252, 16, 251, 199}. Some methods estimate human time with annotation accuracy to jointly model the errors in the annotation process \cite{252, 16, 251}. Other methods vary both the labeling cost and annotation accuracy to calculate a tradeoff between the two \cite{99, 44}. Similarly, some crowdsourcing systems optimize a budget to measure confidence in worker annotations \cite{107, 108}. Models can also predict the redundancy of non-expert labels needed to match expert-level annotations \cite{214}. Just like these methods, we show that non-experts can use our technique and provide expert-quality annotations; we also compare our methods to the conventional majority-voting annotation scheme.

Another perspective on rapid crowdsourcing is to return results in real time, often by using a retainer model to recall workers quickly \cite{7, 131, 128}. Like our approach, real-time crowdsourcing can use algorithmic solutions to combine multiple in-progress contributions \cite{129}. These systems’ techniques could be fused with ours to create crowds that can react to bursty requests.

One common method for optimizing crowdsourcing is active learning, which involves learning algorithms that interactively query the user. Examples include training image recognition \cite{225} and attribution recognition \cite{176} with fewer examples. Comparative models for ranking attribute models have also optimized crowdsourcing using active learning \cite{139}. Similar techniques have explored optimization of the “crowd kernel” by adaptively choosing the next questions asked of the crowd in order to build a similarity matrix between a given set of data points \cite{232}. Active learning needs to decide on a new task after each new piece of data is gathered from the crowd. Such models tend to be quite expensive to compute. Other methods have been proposed to decide on a set of tasks instead of just one task \cite{246}. We draw on this literature: in our technique, after all the images have been seen by at least one worker, we use active learning to decide the next set of tasks. We determine which images to discard and which images to group together and send this set to another worker to
CHAPTER 4. EMBRACING ERROR TO ENABLE RAPID CROWDSOURCING

Figure 4.2: (a) Task instructions inform workers that we expect them to make mistakes since the items will be displayed rapidly. (b) A string of countdown images prepares them for the rate at which items will be displayed. (c) An example image of a “dog” shown in the stream—the two images appearing behind it are included for clarity but are not displayed to workers. (d) When the worker presses a key, we show the last four images below the stream of images to indicate which images might have just been labeled.

gather more information.

Finally, there is a group of techniques that attempt to optimize label collection by reducing the number of questions that must be answered by the crowd. For example, a hierarchy in label distribution can reduce the annotation search space [44], and information gain can reduce the number of labels necessary to build large taxonomies using a crowd [35, 14]. Methods have also been proposed to maximize accuracy of object localization in images [230] and videos [249]. Previous labels can also be used as a prior to optimize acquisition of new types of annotations [15]. One of the benefits of our technique is that it can be used independently of these others to jointly improve crowdsourcing schemes. We demonstrate the gains of such a combination in our evaluation.

4.3 Error-Embracing Crowdsourcing

Current microtask crowdsourcing platforms like Amazon Mechanical Turk incentivize workers to avoid rejections [96, 153], resulting in slow and meticulous work. But is such careful work necessary to build an accurate dataset? In this section, we detail our technique for rapid crowdsourcing by encouraging less accurate work.

The design space of such techniques must consider which tradeoffs are acceptable to make. The first relevant dimension is accuracy. When labeling a large dataset (e.g., building a dataset of ten thousand articles about housing), precision is often the highest priority: articles labeled as on-topic by the system must in fact be about housing. Recall, on the other hand, is often less important, because there is typically a large amount of available unlabeled data: even if the system misses some on-topic articles, the system can label more items until it reaches the desired dataset size. We thus develop an approach for producing high precision at high speed, sacrificing some recall if necessary.
CHAPTER 4. EMBRACING ERROR TO ENABLE RAPID CROWDSOURCING

The second design dimension involves the task characteristics. Many large-scale crowdsourcing tasks involve closed-ended responses such as binary or categorical classifications. These tasks have two useful properties. First, they are time-bound by users’ perception and cognition speed rather than motor (e.g., pointing, typing) speed, since acting requires only a single button press. Second, it is possible to aggregate responses automatically, for example with majority vote. Open-ended crowdsourcing tasks such as writing or transcription are often time-bound by data entry motor speeds and cannot be automatically aggregated. Thus, with our technique, we focus on closed-ended tasks.

4.3.1 Rapid crowdsourcing of binary decision tasks

Binary questions are one of the most common classes of crowdsourcing tasks. Each yes-or-no question gathers a label on whether each item has a certain characteristic. In our technique, rather than letting workers focus on each item too carefully, we display each item for a specific period of time before moving on to the next one in a rapid slideshow. For example, in the context of an image verification task, we show workers a stream of images and ask them to press the spacebar whenever they see a specific class of image. In the example in Figure 4.2, we ask them to react whenever they see a “dog.”

The main parameter in this approach is the length of time each item is visible. To determine the best option, we begin by allowing workers to work at their own pace. This establishes an initial average time period, which we then slowly decrease in successive versions until workers start making mistakes. Once we have identified this error point, we can algorithmically model workers’ latency and errors to extract the true labels.

To avoid stressing out workers, it is important that the task instructions convey the nature of the rapid task and the fact that we expect them to make some errors. Workers are first shown a set of instructions (Figure 4.2(a)) for the task. They are warned that reacting to every single correct image on time is not feasible and thus not expected. We also warn them that we have placed a small number of items in the set that we know to be positive items. These help us calibrate each worker’s speed and also provide us with a mechanism to reject workers who do not react to any of the items.

Once workers start the stream (Figure 4.2(b)), it is important to prepare them for pace of the task. We thus show a film-style countdown for the first few seconds that decrements to zero at the same interval as the main task. Without these countdown images, workers use up the first few seconds getting used to the pace and speed. Figure 4.2(c) shows an example “dog” image that is displayed in front of the user. The dimensions of all items (images) shown are held constant to avoid having to adjust to larger or smaller visual ranges.

When items are displayed for less than 400ms, workers tend to react to all positive items with a delay. If the interface only reacts with a simple confirmation when workers press the spacebar, many workers worry that they are too late because another item is already on the screen. Our solution is
Figure 4.3: Example raw worker outputs from our interface. Each image was displayed for 100ms and workers were asked to react whenever they saw images of “a person riding a motorcycle.” Images are shown in the same order they appeared in for the worker. Positive images are shown with a blue bar below them and users’ keypresses are shown as red bars below the image to which they reacted.

Because of workers’ reaction delay, the data from one worker has considerable uncertainty. We thus show the same set of items to multiple workers in different random orders and collect independent sets of keypresses. This randomization will produce a cleaner signal in aggregate and later allow us to estimate the images to which each worker intended to react.

Given the speed of the images, workers are not able to detect every single positive image. For example, the last positive image in Figure 4.3(a) and the first positive image in Figure 4.3(b) are not detected. Previous work on RSVP found a phenomenon called “attention blink” \cite{18}, in which a
worker is momentarily blind to successive positive images. However, we find that even if two images of “a person riding a motorcycle” occur consecutively, workers are able to detect both and react twice (Figures 4.3(a) and 4.3(b)). If workers are forced to react in intervals of less than 400ms, though, the signal we extract is too noisy for our model to estimate the positive items.

4.3.2 Multi-Class Classification for Categorical Data

So far, we have described how rapid crowdsourcing can be used for binary verification tasks. Now we extend it to handle multi-class classification. Theoretically, all multi-class classification can be broken down into a series of binary verifications. For example, if there are $N$ classes, we can ask $N$ binary questions of whether an item is in each class. Given a list of items, we use our technique to classify them one class at a time. After every iteration, we remove all the positively classified items for a particular class. We use the rest of the items to detect the next class.

Assuming all the classes contain an equal number of items, the order in which we detect classes should not matter. A simple baseline approach would choose a class at random and attempt to detect all items for that class first. However, if the distribution of items is not equal among classes, this method would be inefficient. Consider the case where we are trying to classify items into 10 classes, and one class has 1000 items while all other classes have 10 items. In the worst case, if we classify the class with 1000 examples last, those 1000 images would go through our interface 10 times (once for every class). Instead, if we had detected the large class first, we would be able to classify those 1000 images and they would only go through our interface once. With this intuition, we propose a class-optimized approach that classifies the most common class of items first. We maximize the number of items we classify at every iteration, reducing the total number of binary verifications required.

4.4 Model

To translate workers’ delayed and potentially erroneous actions into identifications of the positive items, we need to model their behavior. We do this by calculating the probability that a particular item is in the positive class given that the user reacted a given period after the item was displayed. By combining these probabilities across several workers with different random orders of the same images, these probabilities sum up to identify the correct items.

We use maximum likelihood estimation to predict the probability of an item being a positive example. Given a set of items $I = \{I_1, \ldots, I_n\}$, we send them to $W$ workers in a different random order for each. From each worker $w$, we collect a set of keypresses $C^w = \{c^w_1, \ldots, c^w_k\}$ where $w \in W$ and $k$ is the total number of keypresses from $w$. Our aim is to calculate the probability of a given item $P(I_i)$ being a positive example. Given that we collect keypresses from $W$ workers:
\[ P(I_i) = \sum w P(I_i | C^w) P(C^w) \] (4.1)

where \( P(C) = \prod_k P(C_k) \) is the probability of a particular set of items being keypresses. We set \( P(C_k) \) to be constant, assuming that it is equally likely that a worker might react to any item. Using Bayes’ rule:

\[ P(I_i | C^w) = \frac{P(C^w | I_i) P(I_i)}{P(C^w)} \] (4.2)

\( P(I_i) \) models our estimate of item \( I_i \) being positive. It can be a constant, or it can be an estimate from a domain-specific machine learning algorithm \[104\]. For example, to calculate \( P(I_i) \), if we were trying to scale up a dataset of “dog” images, we would use a small set of known “dog” images to train a binary classifier and use that to calculate \( P(I_i) \) for all the unknown images. With image tasks, we use a pretrained convolutional neural network to extract image features \[218\] and train a linear support vector machine to calculate \( P(I_i) \).

We model \( P(C^w | I_i) \) as a set of independent keypresses:

\[ P(C^w | I_i) = P(c^w_1, \ldots, c^w_k | I_i) = \prod_k P(C^w_k | I_i). \] (4.3)

Finally, we model each keypress as a Gaussian distribution \( \mathcal{N}(\mu, \sigma) \) given a positive item. We train the mean \( \mu \) and variance \( \sigma \) by running rapid crowdsourcing on a small set of items for which we already know the positive items. Here, the mean and variance of the distribution are modeled to estimate the delays that a worker makes when reacting to a positive item.

Intuitively, the model works by treating each keypress as creating a Gaussian “footprint” of positive probability on the images about 400ms before the keypress (Figure 4.1). The model combines these probabilities across several workers to identify the images with the highest overall probability.

Now that we have a set of probabilities for each item, we need to decide which ones should be classified as positive. We order the set of items \( I \) according to likelihood of being in the positive class \( P(I_i) \). We then set all items above a certain threshold as positive. This threshold is a hyperparameter that can be tuned to trade off precision vs. recall.

In total, this model has two hyperparameters: (1) the threshold above which we classify images as positive and (2) the speed at which items are displayed to the user. We model both hyperparameters in a per-task (image verification, sentiment analysis, etc.) basis. For a new task, we first estimate how long it takes to label each item in the conventional setting with a small set of items. Next, we continuously reduce the time each item is displayed until we reach a point where the model is unable to achieve the same precision as the untimed case.

### 4.5 Calibration: Baseline Worker Reaction Time

Our technique hypothesizes that guiding workers to work quickly and make errors can lead to results that are faster yet with similar precision. We begin evaluating our technique by first studying worker
Figure 4.4: We plot the change in recall as we vary percentage of positive items in a task. We experiment at varying display speeds ranging from 100ms to 500ms. We find that recall is inversely proportional to the rate of positive stimuli and not to the percentage of positive items.

reaction times as we vary the length of time for which each item is displayed. If worker reaction times have a low variance, we accurately model them. Existing work on RSVP estimated that humans usually react about 400ms after being presented with a cue [255, 187]. Similarly, the model human processor [25] estimated that humans perceive, understand and react at least 240ms after a cue. We first measure worker reaction times, then analyze how frequently positive items can be displayed before workers are unable to react to them in time.

Method. We recruited 1,000 workers on Amazon Mechanical Turk with 96% approval rating and over 10,000 tasks submitted. Workers were asked to work on one task at a time. Each task contained a stream of 100 images of polka dot patterns of two different colors. Workers were asked to react by pressing the spacebar whenever they saw an image with polka dots of one of the two colors. Tasks could vary by two variables: the speed at which images were displayed and the percentage of the positively colored images. For a given task, we held the display speed constant. Across multiple tasks, we displayed images for 100ms to 500ms. We studied two variables: reaction time and recall. We measured the reaction time to the positive color across these speeds. To study recall (percentage of positively colored images detected by workers), we varied the ratio of positive images from 5% to 95%. We counted a keypress as a detection only if it occurred within 500ms of displaying a positively colored image.

Results. Workers’ reaction times corresponded well with estimates from previous studies. Workers
CHAPTER 4. EMBRACING ERROR TO ENABLE RAPID CROWDSOURCING

<table>
<thead>
<tr>
<th>Task</th>
<th>Conventional Approach</th>
<th>Our Technique</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Verification</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy</td>
<td>1.50</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>Medium</td>
<td>1.70</td>
<td>0.97</td>
<td>0.99</td>
</tr>
<tr>
<td>Hard</td>
<td>1.90</td>
<td>0.93</td>
<td>0.89</td>
</tr>
<tr>
<td>All</td>
<td>1.70</td>
<td>0.97</td>
<td>0.96</td>
</tr>
<tr>
<td>Sentiment Analysis</td>
<td>4.25</td>
<td>0.93</td>
<td>0.97</td>
</tr>
<tr>
<td>Word Similarity</td>
<td>6.23</td>
<td>0.89</td>
<td>0.94</td>
</tr>
<tr>
<td>Topic Detection</td>
<td>14.33</td>
<td>0.96</td>
<td>0.94</td>
</tr>
</tbody>
</table>

Table 4.1: We compare the conventional approach for binary verification tasks (image verification, sentiment analysis, word similarity and topic detection) with our technique and compute precision and recall scores. Precision scores, recall scores and speedups are calculated using 3 workers in the conventional setting. Image verification, sentiment analysis and word similarity used 5 workers using our technique, while topic detection used only 2 workers. We also show the time taken (in seconds) for 1 worker to do each task.

...tend to react an average of 378ms (σ = 92ms) after seeing a positive image. This consistency is an important result for our model because it assumes that workers have a consistent reaction delay.

As expected, recall is inversely proportional to the speed at which the images are shown. A worker is more likely to miss a positive image at very fast speeds. We also find that recall decreases as we increase the percentage of positive items in the task. To measure the effects of positive frequency on recall, we record the percentage threshold at which recall begins to drop significantly at different speeds and positive frequencies. From Figure 4.4, at 100ms, we see that recall drops when the percentage of positive images is more than 35%. As we increase the time for which an item is displayed, however, we notice that the drop in recall occurs at a much higher percentage. At 500ms, the recall drops at a threshold of 85%. We thus infer that recall is inversely proportional to the rate of positive stimuli and not to the percentage of positive images. From these results we conclude that at faster speeds, it is important to maintain a smaller percentage of positive images, while at slower speeds, the percentage of positive images has a lesser impact on recall. Quantitatively, to maintain a recall higher than 0.7, it is necessary to limit the frequency of positive cues to one every 400ms.

4.6 Study 1: Image Verification

In this study, we deploy our technique on image verification tasks and measure its speed relative to the conventional self-paced approach. Many crowdsourcing tasks in computer vision require verifying that a particular image contains a specific class or concept. We measure precision, recall and cost (in seconds) by the conventional approach and compare against our technique.

Some visual concepts are easier to detect than others. For example, detecting an image of a “dog”
CHAPTER 4. EMBRACING ERROR TO ENABLE RAPID CROWDSOURCING

Figure 4.5: We study the precision (left) and recall (right) curves for detecting “dog” (top), “a person on a motorcycle” (middle) and “eating breakfast” (bottom) images with a redundancy ranging from 1 to 5. There are 500 ground truth positive images in each experiment. We find that our technique works for simple as well as hard concepts.

is a lot easier than detecting an image of “a person riding a motorcycle” or “eating breakfast.” While detecting a “dog” is a perceptual task, “a person riding a motorcycle” requires understanding of the interaction between the person and the motorcycle. Similarly, “eating breakfast” requires workers to fuse concepts of people eating a variety foods like eggs, cereal or pancakes. We test our technique on detecting three concepts: “dog” (easy concept), “a person riding a motorcycle” (medium concept) and “eating breakfast” (hard concept). In this study, we compare how workers fare on each of these three levels of concepts.

Method. In this study, we compare the conventional approach with our technique on three (easy, medium and hard) concepts. We evaluate each of these comparisons using precision scores, recall scores and the speedup achieved. To test each of the three concepts, we labeled 10,000 images, where each concept had 500 examples. We divided the 10,000 images into streams of 100 images for each task. We paid workers $0.17 to label a stream of 100 images (resulting in a wage of $6 per hour [205]). We hired over 1,000 workers for this study satisfying the same qualifications as the calibration task.

The conventional method of collecting binary labels is to present a crowd worker with a set of items. The worker proceeds to label each item, one at a time. Most datasets employ multiple workers to label each task because majority voting [222] has been shown to improve the quality of crowd
annotations. These datasets usually use a redundancy of 3 to 5 workers. In all our experiments, we used a redundancy of 3 workers as our baseline.

When launching tasks using our technique, we tuned the image display speed to 100ms. We used a redundancy of 5 workers when measuring precision and recall scores. To calculate speedup, we compare the total worker time taken by all the 5 workers using our technique with the total worker time taken by the 3 workers using the conventional method. Additionally, we vary redundancy on all the concepts from 1 to 10 workers to see its effects on precision and recall.

Results. Self-paced workers take 1.70s on average to label each image with a concept in the conventional approach (Table 4.1). They are quicker at labeling the easy concept (1.50s per worker) while taking longer on the medium (1.70s) and hard (1.90s) concepts.

Using our technique, even with a redundancy of 5 workers, we achieve a speedup of 10.20× across all concepts. We achieve order of magnitude speedups of 9.00×, 10.20× and 11.40× on the easy, medium and hard concepts. Overall, across all concepts, the precision and recall achieved by our technique is 0.97 and 0.81. Meanwhile the precision and recall of the conventional method is 0.97 and 0.96. We thus achieve the same precision as the conventional method. As expected, recall is lower because workers are not able to detect every single true positive example. As argued previously, lower recall can be an acceptable tradeoff when it is easy to find more unlabeled images.

Now, let’s compare precision and recall scores between the three concepts. We show precision and recall scores in Figure 4.5 for the three concepts. Workers perform slightly better at finding “dog” images and find it the most difficult to detect the more challenging “eating breakfast” concept. With a redundancy of 5, the three concepts achieve a precision of 0.99, 0.98 and 0.90 respectively at a recall of 0.94, 0.83 and 0.74 (Table 4.1). The precision for these three concepts are identical to the conventional approach, while the recall scores are slightly lower. The recall for a more difficult cognitive concept (“eating breakfast”) is much lower, at 0.74, than for the other two concepts. More complex concepts usually tend to have a lot of contextual variance. For example, “eating breakfast” might include a person eating a “banana,” a “bowl of cereal,” “waffles” or “eggs.” We find that while some workers react to one variety of the concept (e.g., “bowl of cereal”), others react to another variety (e.g., “eggs”).

When we increase the redundancy of workers to 10 (Figure 4.6), our model is able to better approximate the positive images. We see diminishing increases in both recall and precision as redundancy increases. At a redundancy of 10, we increase recall to the same amount as the conventional approach (0.96), while maintaining a high precision (0.99) and still achieving a speedup of 5.1×.

We conclude from this study that our technique (with a redundancy of 5) can speed up image verification with easy, medium and hard concepts by an order of magnitude while still maintaining high precision. We also show that recall can be compensated by increasing redundancy.
CHAPTER 4. EMBRACING ERROR TO ENABLE RAPID CROWDSOURCING

Figure 4.6: We study the effects of redundancy on recall by plotting precision and recall curves for detecting “a person on a motorcycle” images with a redundancy ranging from 1 to 10. We see diminishing increases in precision and recall as we increase redundancy. We manage to achieve the same precision and recall scores as the conventional approach with a redundancy of 10 while still achieving a speedup of $5 \times$.

4.7 Study 2: Non-Visual Tasks

So far, we have shown that rapid crowdsourcing can be used to collect image verification labels. We next test the technique on a variety of other common crowdsourcing tasks: sentiment analysis [174], word similarity [222] and topic detection [136].

Method. In this study, we measure precision, recall and speedup achieved by our technique over the conventional approach. To determine the stream speed for each task, we followed the prescribed method of running trials and speeding up the stream until the model starts losing precision. For sentiment analysis, workers were shown a stream of tweets and asked to react whenever they saw a positive tweet. We displayed tweets at 250ms with a redundancy of 5 workers. For word similarity, workers were shown a word (e.g., “lad”) for which we wanted synonyms. They were then rapidly shown other words at 600ms and asked to react if they see a synonym (e.g., “boy”). Finally, for topic detection, we presented workers with a topic like “housing” or “gas” and presented articles of an average length of 105 words at a speed of 2s per article. They reacted whenever they saw an article containing the topic we were looking for. For all three of these tasks, we compare precision, recall and speed against the self-paced conventional approach with a redundancy of 3 workers. Every task, for both the conventional approach and our technique, contained 100 items.

To measure the cognitive load on workers for labeling so many items at once, we ran the widely-used NASA Task Load Index (TLX) [37] on all tasks, including image verification. TLX measures the perceived workload of a task. We ran the survey on 100 workers who used the conventional approach and 100 workers who used our technique across all tasks.

Results. We present our results in Table [4.1] and Figure [4.7]. For sentiment analysis, we find that workers in the conventional approach classify tweets in 4.25s. So, with a redundancy of 3 workers, the conventional approach would take 12.75s with a precision of 0.93. Using our method and a
redundancy of 5 workers, we complete the task in 1250ms (250ms per worker per item) and 0.94 precision. Therefore, our technique achieves a speedup of 10.2×.

Likewise, for word similarity, workers take around 6.23s to complete the conventional task, while our technique succeeds at 600ms. We manage to capture a comparable precision of 0.88 using 5 workers against a precision of 0.89 in the conventional method with 3 workers. Since finding synonyms is a higher-level cognitive task, workers take longer to do word similarity tasks than image verification and sentiment analysis tasks. We manage a speedup of 6.23×.

Finally, for topic detection, workers spend significant time analyzing articles in the conventional setting (14.33s on average). With 3 workers, the conventional approach takes 43s. In comparison, our technique delegates 2s for each article. With a redundancy of only 2 workers, we achieve a precision of 0.95, similar to the 0.96 achieved by the conventional approach. The total worker time to label one article using our technique is 4s, a speedup of 10.75×.

The mean TLX workload for the control condition was 58.5 (σ = 9.3), and 62.4 (σ = 18.5) for our technique. Unexpectedly, the difference between conditions was not significant (t(99) = −0.53, p = 0.59). The temporal demand scale item appeared to be elevated for our technique (61.1 vs. 70.0), but this difference was not significant (t(99) = −0.76, p = 0.45). We conclude that our technique can be used to scale crowdsourcing on a variety of tasks without statistically increasing worker workload.

4.8 Study 3: Multi-class Classification

In this study, we extend our technique from binary to multi-class classification to capture an even larger set of crowdsourcing tasks. We use our technique to create a dataset where each image is classified into one category (“people,” “dog,” “horse,” “cat,” etc.). We compare our technique with a conventional technique [43] that collects binary labels for each image for every single possible class.

Method. Our aim is to classify a dataset of 2,000 images with 10 categories where each category contains between 100 to 250 examples. We compared three methods of multi-class classification: (1) a naive approach that collected 10 binary labels (one for each class) for each image, (2) a baseline approach that used our interface and classified images one class (chosen randomly) at a time, and (3) a class-optimized approach that used our interface to classify images starting from the class with the most examples. When using our interface, we broke tasks into streams of 100 images displayed for 100ms each. We used a redundancy of 3 workers for the conventional interface and 5 workers for our interface. We calculated the precision and recall scores across each of these three methods as well as the cost (in seconds) of each method.

Results. (1) In the naive approach, we need to collect 20,000 binary labels that take 1.7s each. With 5 workers, this takes 102,000s ($170 at a wage rate of $6/hr) with an average precision of 0.99 and recall of 0.95. (2) Using the baseline approach, it takes 12,342s ($20.57) with an average precision of 0.98 and recall of 0.83. This shows that the baseline approach achieves a speedup of
8.26× when compared with the naive approach. (3) Finally, the class-optimized approach is able to detect the most common class first and hence reduces the number of times an image is sent through our interface. It takes 11,700s ($19.50) with an average precision of 0.98 and recall of 0.83. The class-optimized approach achieves a speedup of 8.7× when compared to the naive approach. While the speedup between the baseline and the class-optimized methods is small, it would be increased on a larger dataset with more classes.

4.9 Application: Building ImageNet

Our method can be combined with existing techniques [44, 225, 176, 11] that optimize binary verification and multi-class classification by preprocessing data or using active learning. One such method [44] annotated ImageNet (a popular large dataset for image classification) effectively with a useful insight: they realized that its classes could be grouped together into higher semantic concepts. For example, “dog,” “rabbit” and “cat” could be grouped into the concept “animal.” By utilizing the hierarchy of labels that is specific to this task, they were able to preprocess and reduce the number of labels needed to classify all images. As a case study, we combine our technique with their insight...
and evaluate the speedup in collecting a subset of ImageNet.

Method. We focused on a subset of the dataset with 20,000 images and classified them into 200 classes. We conducted this case study by comparing three ways of collecting labels: (1) The naive approach asked 200 binary questions for each image in the subset, where each question asked if the image belonged to one of the 200 classes. We used a redundancy of 3 workers for this task. (2) The optimal-labeling method used the insight to reduce the number of labels by utilizing the hierarchy of image classes. (3) The combined approach used our technique for multi-class classification combined with the hierarchy insight to reduce the number of labels collected. We used a redundancy of 5 workers for this technique with tasks of 100 images displayed at 250ms.

Results. (1) Using the naive approach, this would result in asking 4 million binary verification questions. Given that each binary label takes 1.7s (Table 4.1), we estimate that the total time to label the entire dataset would take 6.8 million seconds ($11,333 at a wage rate of $6/hr). (2) The optimal-labeling method is estimated to take 1.13 million seconds ($1,888) [44]. (3) Combining the hierarchical questions with our interface, we annotate the subset in 136,800s ($228). We achieve a precision of 0.97 with a recall of 0.82. By combining our 8× speedup with the 6× speedup from intelligent question selection, we achieve a 50× speedup in total.

4.10 Discussion

We focused our technique on positively identifying concepts. We then also test its effectiveness at classifying the absence of a concept. Instead of asking workers to react when they see a “dog,” if we ask them to react when they do not see a “dog,” our technique performs poorly. At 100ms, we find that workers achieve a recall of only 0.31, which is much lower than a recall of 0.94 when detecting the presence of “dog”s. To improve recall to 0.90, we must slow down the feed to 500ms. Our technique achieves a speedup of 2× with this speed. We conclude that our technique performs poorly for anomaly detection tasks, where the presence of a concept is common but its absence, an anomaly, is rare. More generally, this exercise suggests that some cognitive tasks are less robust to rapid judgments. Preattentive processing can help us find “dog”s, but ensuring that there is no “dog” requires a linear scan of the entire image.

To better understand the active mechanism behind our technique, we turn to concept typicality. A recent study [93] used fMRIs to measure humans’ recognition speed for different object categories, finding that images of most typical exemplars from a class were recognized faster than the least typical categories. They calculated typicality scores for a set of image classes based on how quickly humans recognized them. In our image verification task, 72% of false negatives were also atypical. Not detecting atypical images might lead to the curation of image datasets that are biased towards more common categories. For example, when curating a dataset of dogs, our technique would be more likely to find usual breeds like “dalmatians” and “labradors” and miss rare breeds like “romagnolos”
and “otterhounds.” More generally, this approach may amplify biases and minimize clarity on edge cases. Slowing down the feed reduces atypical false negatives, resulting in a smaller speedup but with a higher recall for atypical images.

4.11 Conclusion

We have suggested that crowdsourcing can speed up labeling by encouraging a small amount of error rather than forcing workers to avoid it. We introduce a rapid slideshow interface where items are shown too quickly for workers to get all items correct. We algorithmically model worker errors and recover their intended labels. This interface can be used for binary verification tasks like image verification, sentiment analysis, word similarity and topic detection, achieving speedups of $10.2\times$, $10.2\times$, $6.23\times$ and $10.75\times$ respectively. It can also extend to multi-class classification and achieve a speedup of $8.26\times$. Our approach is only one possible interface instantiation of the concept of encouraging some error; we suggest that future work may investigate many others. Speeding up crowdsourcing enables us to build larger datasets to empower scientific insights and industry practice. For many labeling goals, this technique can be used to construct datasets that are an order of magnitude larger without increasing cost.

4.12 Supplementary Material

4.12.1 Runtime Analysis for Class-Optimized Classification

In the paper, we show how our interface can be used for multi-class classification. We also compared a baseline approach with a class-optimized approach where we detect classes in decreasing order of the number of example items it has. We provided a case for why the class-optimized approach performs better. In this section, we provide a run time analysis of the two approaches.

Let’s consider the case where we have $M$ classes and each class has $N_i$ examples where $i \in M$. Let the class with the most number of examples contain $N_{\text{max}}$ items such that:

$$N_{\text{max}} = \max_i N_i$$

(4.4)

We make the assumption that $N_{\text{max}} \gg M$, i.e. the number of examples of at least one class is much larger than the total number of classes.

Consider the baseline approach where we pick classes to detect in a random order. In the worst case, we choose classes such that the class with the most number of examples is chosen last. In this case, these $N_{\text{max}}$ images have gone through our interface once for every class, resulting in a runtime of $O(M \cdot N_{\text{max}})$.

The runtime for this can be improved by using the class-optimized approach where we classify objects into classes in decreasing number of positive examples. In this case, the $N_{\text{max}}$ objects go
through our interface at the very beginning only once and get classified. Assuming $N_{\text{max}} \gg N_j \forall j \in M$, this results in a runtime of $O(N_{\text{max}})$. We conclude that the class-optimized approach achieves linear time versus quadratic.
Chapter 5

Long-Term Crowd Worker Quality

5.1 Introduction

Microtask crowdsourcing is gaining popularity among corporate and research communities as a means to leverage parallel human computation for extremely large problems [233, 8, 7, 131, 129]. These communities use crowd work to complete hundreds of thousands of tasks per day [151], from which new datasets with over 20 million annotations can be produced within a few months [118]. A crowdsourcing platform like Amazon’s Mechanical Turk (AMT) is a marketplace subject to human factors that affect its performance, both in terms of speed and quality [47]. Prior studies found that work division in crowdsourcing follows a Pareto principle, where a small minority of workers usually completes a great majority of the work [141]. If such large crowdsourced projects are being completed by a small percentage of workers, then these workers spend hours, days, or weeks executing the exact same tasks. Consequently, we pose the question:

*How does a worker’s quality change over time?*

Multiple arguments from previous literature in psychology suggest that quality should decrease over time. *Fatigue*, a temporary decline in cognitive or physical condition, can gradually result in performance drops over long periods of time [179, 13, 122]. Since the microtask paradigm in large scale crowdsourcing involves monotonous sequences of repetitive tasks, fatigue buildup can pose a potential problem to the quality of submitted work over time [39]. Furthermore, workers have been noted to be “satisficers” who, as they gain familiarity with the task and its acceptance thresholds, strive to do the minimal work possible to achieve these thresholds [217, 27].

To study these long term effects on crowd work, we analyze worker trends over three different real-world, large-scale datasets [118] collected from microtasks on AMT: image descriptions, question answering, and binary verifications. With microtasks comprising over 60% of the total crowd work and microtasks involving images being the most common type [87], these datasets cover a large
percentage of the type of crowd work most commonly seen. Specifically, we use over 5 million image
descriptions from 2674 workers over a 9 month span, 0.8 million question-answer pairs from 2179
workers over a 3 month span, and 2 million verifications from 3913 workers over a 3 month span.
The average worker in the largest dataset worked for an average of 2 eight-hour work days while
the top 1% of workers worked for nearly 45 eight-hour work days. Using these datasets, we look at
temporal trends in the accuracy of annotations from workers, diversity of these annotations, and the
speed of completion.

Contrary to our hypothesis that workers would exhibit glaring signs of fatigue via large declines
in submission quality over time, we find that workers who complete large sets of microtasks maintain
a consistent level of quality (measured as the percentage of correct annotations). Furthermore, as
workers become more experienced on a task, they develop stable strategies that do not change,
enabling them to complete tasks faster. But are workers generally consistent or is this consistency
simply a product of the task design?

We thus perform an experiment where we hire workers from AMT to complete large-scale tasks
while randomly assigning them into different task designs. These designs were varied across two
factors: the acceptance threshold with which we accept or reject work, and the transparency of that
threshold. If workers manipulate their quality level strategically to avoid rejection, workers with
a high (difficult) threshold would perform at a noticeably better level than the ones with a low
threshold who can satisfice more aggressively. However, this effect might only be easily visible if
workers have transparency into how they performed on the task.

By analyzing 676,628 annotations collected from 1134 workers in the experiment on AMT, we
found that workers display consistent quality regardless of their assigned condition, and that lower-
quality workers in the high threshold condition would often self-select out of tasks where they believe
there is a high risk of rejection. Bolstered by this consistency, we ask: can we predict a worker’s
future quality months after they start working on a microtask?

If individual workers indeed sustain constant correctness over time, then, intuitively, any subset of
a worker’s submissions should be representative of their entire work. We demonstrate that a simple
glimpse of a worker’s quality in their first few tasks is a strong predictor of their long-term quality.
Simply averaging the quality of work of a worker’s first 5 completed tasks can predict that worker’s
quality during the final 10% of their completed tasks with an average error of 3.4%.

Long-term worker consistency suggests that paying attention to easy signals of good workers can
be key to collecting a large dataset of high quality annotations [162, 202]. Once we have identified
these workers, we can back off the gold-standard (attention check) questions to ensure good quality
work, since work quality is unvarying [142]. We can also be more permissive about errors from
workers known to be good, reducing the rejection risk that workers face and increasing worker
retention [46, 133].
5.2 Related Work

Our work is inspired by psychology, decision making, and workplace management literature that focuses on identifying the major factors that affect the quality of work produced. Specifically, we look at the effects of fatigue and satisficing in the workplace. We then study whether these problems transfer to the crowdsourcing domain. Next, we explore how our contributions are necessary to better understand the global ecosystem of crowdsourcing. Finally, we discuss the efficacy of existing worker quality improvement techniques.

5.2.1 Fatigue

Repeatedly completing the same task over a sustained period of time will induce fatigue, which increases reaction time, decreases production rate, and is linked to a rise in poor decision-making \[122, 260\]. The United States Air Force found that both the cognitive performance and physical conditions of its airmen continually deteriorated during the course of long, mandatory shifts \[179\]. However, unlike these mandatory, sustained shifts, crowdsourcing is generally opt-in for workers — there always exists the option for workers to break or find another task whenever they feel tired or bored \[130, 132\]. Nonetheless, previous work has shown that people cannot accurately gauge how long they need to rest after working continuously, resulting in incomplete recoveries and drops in task performance after breaks \[86\]. Ultimately, previous work in fatigue suggests that crowd workers who continuously complete tasks over sustained periods would result in significant decreases in work quality. We show that contrary to this literature, crowd workers remain consistent throughout their time on a specific task.

5.2.2 Satisficing

Crowd workers are often regarded as “satisficers” who do the minimal work needed for their work to be accepted \[217, 27\]. Examples of satisficing in crowdsourcing occur during surveys \[121\] and when workers avoid the most difficult parts of a task \[155\]. Disguised attention checks in the instructions \[169\] or rate-limiting the presentation of the questions \[105\] improves the detection and prevention of satisficing. Previous studies of crowd workers’ perspectives find that crowd workers believe themselves to be genuine workers, monitoring their own work and giving helpful feedback to requesters \[156\]. Workers have also been shown to respond well and produce high quality work if the task is designed to be effort-responsive \[88\]. However, workers often consider the cost-benefit of continuing to work on a particular task — if they feel that a task is too time-consuming relative to its reward, then they often drop out or compensate by satisficing (e.g. reducing quality) \[156\]. Prior work has shown that We observe that satisficing does occur, but it only affects a small portion of long-term workers. We also observe in our experiments that workers opt out of tasks where they feel they have a high risk of rejection.
5.2.3 The global crowdsourcing ecosystem

With the rapidly growing size of crowdsourcing projects, workers now have the opportunity to undertake large batches of tasks. As they progress through these tasks, questions arise and they often seek help by communicating with other workers or the task creator \[153\]. Furthermore, on external forums and in collectives, workers often share well-paying work opportunities, teach and learn from other workers, review requesters, and even consult with task creators to give constructive feedback \[153, 96, 205, 156\]. When considering this crowdsourcing ecosystem, crowd researchers often envision how more complex workflows can be integrated to make the overall system more efficient, fair, and allow for a wider range of tasks to be possible \[113\]. To continue the trend towards a more complex, but more powerful, crowdsourcing ecosystem, it is imperative that we study the long-term trends of how workers operate within it. Our paper seeks to identify trends that occur as workers continually complete tasks over a long period of time. We conclude that crowdsourcing workflows should design methods to identify good workers and provide them with the ability to complete tasks with a low threshold for acceptance as good workers work consistently hard regardless of the acceptance criteria.

5.2.4 Improving crowdsourcing quality

External checks such as verifiable gold standards, requiring explanations, and majority voting are standard practice for reducing bad answers and quality control \[112, 24\]. Other methods directly estimate worker quality to improve these external checks \[95, 257\]. Giving external feedback or having crowd workers internally reflect on their prior work also has been shown to yield better results \[50\]. Previous work directly targets the monotony of crowdsourcing, showing that by framing the task as more meaningful to workers (for example as a charitable cause), one obtains higher
CHAPTER 5. LONG-TERM CROWD WORKER QUALITY

quality results [26]. However, this framing study only had workers do each task a few times and did not observe long-term trends. We, on the other hand, explore the changes in worker quality on microtasks that are repeated by workers over long periods of time.

5.3 Analysis: Long-Term Crowdsourcing Trends

In this section, we perform an analysis of worker behavior over time on large-scale datasets of three machine learning labeling tasks: image descriptions, question answering, and binary verification. We examine common trends, such as worker accuracy and annotation diversity over time. We then use our results to answer whether workers are fatiguing or displaying other decreases in effectiveness over time.

5.3.1 Data

We first describe the three datasets that we inspect. Each of the three tasks were priced such that workers could earn $6 per hour and were only available to workers with a 95% approval rating and who live in the United States. For the studies in this paper, workers were tracked by their AMT worker ID’s. The tasks and interfaces used to collect the data are described in further detail in the Visual Genome paper [118].

Image descriptions. An image description is a phrase or sentence associated with a certain part of an image. To complete this task, a worker looks at an image, clicks and drags to select an area of the image, and then describes it using a short textual phrase (e.g., “The dog is jumping to catch the frisbee”). Each image description task requires a worker to create $5 - 10$ unique descriptions for one randomly selected image, averaging at least 5 words per description. Workers were asked to keep the descriptions factual and avoid submitting any speculative phrases or sentences. We estimate that each task takes around 4 minutes and we allotted 2.5 hours such that workers did not feel pressured for time. In total, 5,380,263 image descriptions were collected from 2674 workers over 9 months.

Question answers. Each question answering task asks a worker to write 7 questions and their corresponding answers per image for 2 different, randomly selected images. Workers were instructed to begin each sentence with one of the following questions: who, what, when, where, why and how [124]. Furthermore, to ensure diversity of question types, workers were asked to write a minimum of 4 of these question types. Workers were also instructed to be concise and unambiguous to avoid wordy and speculative questions. Each task takes around 4 minutes and we allotted 2.5 hours such that workers did not feel pressured for time. In total, 832,880 question-answer pairs were generated by 2179 workers over 3 months.

Binary verifications. Verification tasks were quality control tasks: given an image and a question-answer pair, workers were asked if the question was relevant to the image and if the answer accurately responded to the question. The majority decision of 3 workers was used to determine the accuracy of
Table 5.1: The number of workers, tasks, and annotations collected for image descriptions, question answering, and verifications.

<table>
<thead>
<tr>
<th></th>
<th>Annotations</th>
<th>Tasks</th>
<th>Workers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descriptions</td>
<td>5,380,263</td>
<td>605,443</td>
<td>2674</td>
</tr>
<tr>
<td>Question Answering</td>
<td>830,625</td>
<td>54,587</td>
<td>2179</td>
</tr>
<tr>
<td>Verification</td>
<td>2,689,350</td>
<td>53,787</td>
<td>3913</td>
</tr>
</tbody>
</table>

Each question answering pair. For each verification task, a worker voted on 50 randomly-ordered question-answer pairs. Each task takes around 3 minutes and we allotted 1 hour such that workers did not feel pressured for time. In total, 2,498,640 votes were cast by 3913 workers over 3 months.

Overall. Figure 5.1 shows the distribution of how many tasks workers completed over the span of the data collection period, while Table 5.1 outlines the total number of annotations and tasks completed. The top 20% of workers who completed the most tasks did 91.8%, 90.9%, and 88.9% of the total work in each of the three datasets respectively. These distributions are similar to the standard Pareto 80-20 rule [141], clearly demonstrating that a small, but persistent minority of workers completes an extremely large number of similar tasks. We noticed that workers in the top 1% each completed approximately 1% of the respective datasets each, with 5455 image description tasks, 758 question answering tasks, and 1018 verification tasks completed on average. If each of these workers in the top 1% each took 4 minutes for image descriptions and question answering tasks and 3 minutes for verification tasks, the estimated average work time equates to 45, 6.2 and 6.2 eight-hour work days for each task respectively. This sheer workload demonstrates that workers may work for very extended periods of time on the same task. Additionally, workers, on average, completed at least one task per week for 6 weeks. By the final week of the data collection, about 10% of the workers remained working on the tasks, suggesting that our study captures the entire lifetime of many of these workers.

We focus our attention on workers who completed at least 100 tasks during the span of the data collection. The completion time for 100 tasks is approximately 6.7 hours for image description and question answering tasks and 5.0 hours for verification tasks. We find that 657, 128, and 177 workers completed 100 of the image description, question answering, and verification tasks respectively. The median worker in each task type completed 349, 220, and 181 tasks, which translates to 23.2, 14.6, and 6.0 hours of continuous work. These workers also produced 94.5%, 70.5%, and 66.3% of each of the total annotations. These worker pools are relatively unique: there are 61 shared workers between image descriptions and QA, 69 shared workers between image description and verification, 42 shared workers between question answering and verifications, and 25 shared workers between all three tasks.

We reached out to the 815 unique workers who had worked on at least 100 tasks and asked them to complete a survey. After collecting 305 responses, we found the gender distribution to be 72.8% female, 26.9% male, and 0.3% other (Figure 5.2). Furthermore, we found that workers with ages 30-49 were the majority at 54.1% of the long-term worker population. Ages 18-29, 50-64, and 65+
respectively comprised 19.0%, 23.3% and 3.6% of the long-term worker population. Compared to the distributions in previously gathered demographics on AMT [87, 46, 196], the gender and age distribution of all workers closely aligns with these other previously gathered distributions [118]. However, the distribution of long-term workers is skewed towards older and female workers.

### 5.3.2 Workers are consistent over long periods

We analyzed worker accuracy and annotation diversity over the entire period of time that they worked on these tasks. Because workers performed different numbers of tasks, we normalize time data to percentages of their total lifetime, which we define as the period from when a worker starts the task until they stop working on that task. For example, if one worker completed 200 tasks and another completed 400 tasks, then the halfway point in their respective lifetimes would be when they completed 100 and 200 tasks.

**Annotation accuracy.** A straightforward metric of quality is the percentage of microtasks that are correct. To determine accuracy for an image description or question answering task, we computed the percentage of descriptions or question-answer pairs deemed true by a majority vote made by other workers. However, to use this majority vote in a metric, we need to first validate that this verification process is repeatable and accurate. Since the ground truth of verification tasks is unknown at such a large scale, we need a method to estimate the accuracy of each verification decision. We believe that comparing a worker’s vote against the majority decision is a good approximation of accuracy. To test accuracy, we randomly sampled a set of 1,000 descriptions and image answers and manually compared our own verifications against the majority vote, which resulted in a 98.2% match. To test repeatability, we randomly sampled a set of 15,000 descriptions and question answers to be sent back to be voted on by 3 new workers 6 months after the initial dataset was collected. Ultimately,
we found a 99.3% similarity between the majority decision of this new verification process with the original decision reported in the dataset [118]. The result of this test indicates that the majority decision is both accurate and repeatable, making it a good standard to compare against.

We find that workers change very little over time (Figures 5.3 and 5.4). When considering those who did at least 100 image description tasks, people on average started at 97.9 ± 12.1% accuracy and ended at 96.6 ± 9.1%, averaging an absolute change of 3.3 ± 5.6%. Workers who did at least 100 question answering tasks started with an average of 88.4 ± 6.3% and ended at 87.5 ± 6.0%, resulting in an absolute change of 3.1 ± 3.3%. For the verification task, workers agreed with the majority on average 88.1 ± 3.6% at the start and 89.0 ± 4.0% at the end, resulting in an absolute change of 3.1 ± 3.4%.

**Annotation diversity.** Accuracy captures clearly correct or incorrect outcomes, but how about subtler signals of effort level? Since each image description or question answering task produces multiple phrases or questions, we examine the linguistic similarity of these phrases and questions
over time. As N-grams have often been used in language processing for gauging similarity between documents [40], we construct a metric of syntax diversity for a set of annotations as follows:

\[
\text{diversity} = \frac{\text{number of unique N-grams}}{\text{number of total N-grams}}
\]

(5.1)

As the annotation set increasingly contains different words and ordering of words, this diversity metric approaches 1 because the number of unique N-grams will approach the total possible N-grams. Conversely, if the annotation set contains increasingly similar annotations, many N-grams will be redundant, making this diversity metric approach 0. To account for workers reusing similar sentence structure in consecutive tasks, we track the number of unique N-grams versus total N-grams in sequential pairs of tasks.

Figure 5.5 illustrates that the percentage of unique bigrams decreases slightly over time. In the image description task, the percent of unique bigrams decreases on average from 82.4% to 78.4% between the start and end of a worker’s lifetime. Since there are 4.2 bigrams on average per phrase, a worker writes approximately 42 total bigrams per task. Thus, a decrease in 4.0% results in a loss of 1.7 unique bigrams per task. In the question answering task, the percent of unique bigrams decreases on average from 60.7% to 54.0%. As there are on average 3.4 bigrams per question, this 6.7% decrease would cost a loss of 3.2 distinct bigrams per task. Ultimately, these results show that over the course of a worker’s lifetime, only a small fraction of diversity is lost, as less than a sentence or question’s contribution of bigrams is lost.

A majority of workers stay constant during their lifetime. However, a few workers decrease to an extremely low N-gram diversity, despite writing factually correct image descriptions and questions. This behavior describes a “satisficing” worker, as they repeatedly write the same types of sentences or questions that generalize to almost every image. Figure 5.6 demonstrates how a satisficing worker’s phrase diversity decreases from image-specific descriptions submitted in early-lifetime tasks to generic,
repeated sentences submitted in late-lifetime tasks. To determine the percentage of total workers who are satisficing workers, we first compute the average diversity of submissions for each worker. We then set a threshold equal the difference between the maximum and mean of these diversities, labeling workers below the mean by this threshold as satisficers. We find that approximately 7% and 2% of workers satisfice in the image description and question answering datasets respectively.

**Annotation speed.** We recorded the time it takes on average for workers to complete a single verification. We removed 2.4% of the data points deemed as outliers from this computation, as workers will infrequently take longer times during a break or while reading the instructions. We defined outliers for each task of 50 verifications as times that outside 3 standard deviations of the mean time for those 50 verifications. Overall, Figure 5.7 demonstrates that workers indeed get faster over time. Initially, workers start off taking 4.5 seconds per verification task, but end up averaging under 3.4 seconds per task, resulting in an approximate 25% speedup. Although no time data was recorded for either the image descriptions or question answering tasks, we believe that they would also exhibit similar speedups over time due to practice effects [164] and similarities in the correctness and diversity metrics.
5.3.3 Discussion

No significant fatigue effects are exhibited by long-term workers. Workers do not appear to suffer from long-term fatigue effects. With an insignificant average accuracy drop of on average 1.5% for workers across their lifetime, we find that workers demonstrate little change in their submission quality. Instead of suffering from fatigue, workers may be opting out or breaking whenever they feel tired [39]. Furthermore, this finding agrees with previous literature that cumulative fatigue is not a major factor in quality drop [179].

Accuracy is constant within a task type, but varies across different task types. We attribute the similarity between the average accuracy of the question answering and verification tasks to their sequential relationship in the crowdsourcing pipeline. If the question-answer pairs are ambiguous or speculative, then the majority vote often becomes split, resulting in accuracy loss for both the question answering and verification tasks. Additionally, we notice the average accuracy for image descriptions is noticeably higher than the average accuracy for either the question answering or verification datasets. We believe this discrepancy stems from the question answering task’s instructions that ask workers to write at four distinct types of W questions (e.g. “why”, “what”, “when”). Some question types such as “why” or “when” are often ambiguous for many images (e.g. “why is the man angry?”). Such questions are often marked as incorrect by other workers in the verification task. Furthermore, we also attribute the disparity between unique bigram percentage for the image description and question answering tasks to the question answering task’s instructions that asked workers to begin each question with one of the 7 question types.

Experience translates to efficiency. Workers retain constant accuracy, and slightly reduce the complexity of their writing style. Combined, these findings suggest that workers find a general strategy that leads to acceptance and stick with it. Studies of practice effects suggest that a practiced strategy helps to increase worker throughput according to a power law [164]. This power law shape is clearly evident in the average verification speed, confirming that practice plays a crucial role in the worker speedup.

Overall findings. From an analysis of the three datasets, we found that fatigue effects are not significantly visible and that severe satisficing behavior only affects a very small proportion of workers. On average, workers maintain a similar quality of work over time, but also get more efficient as they gain experience with the task.

5.4 Experiment: Why Are Workers Consistent?

Examining the image descriptions, question answering, and the verification datasets, we find that worker’s performance on a given microtask remains consistent — even if they do the task for multiple months. However, mere observation of this consistency does not give true insight into the reasons for its existence. Thus, we seek to answer the following question: do crowd workers satisfice according
to the minimum quality necessary to get paid, or are they consistent regardless of this minimum quality?

To answer this question, we perform an experiment where we vary the quality threshold of work and the threshold’s visibility. If workers are stable, we would expect them to either submit work that is above or below the threshold, irrespective of what the threshold is. However, if workers satisfice according to the minimum quality expected, they would adjust the quality of their work based on set threshold [217, 121].

If workers indeed satisfice, then the knowledge of this threshold and their own performance should make it easier to perfect satisficing strategies. Therefore, to adequately study the effects of satisficing, we vary the visibility of the threshold to workers as well. In one condition, we display workers’ current quality scores and the minimum quality score to be accepted, while the other condition only displays whether submitted work was accepted or rejected. To sum up, we vary the threshold and the transparency of this threshold to determine how crowd workers react to the same task, but with different acceptability criteria.

5.4.1 Task

To study why workers are consistent, we designed a task where workers are presented with a series of randomly ordered 58 binary verification questions. Each verification requires them to determine if an image description and its associated image part are correct. For example, in Figure 5.8 workers must decide if “the zebras have stripes” is a good description of a particular part of the image. They are asked to base their response based solely on the content of the image and the semantics of the sentence. To keep the task simple, we asked workers to ignore whether the box was perfectly
surrounding the image area being described. The tasks were priced such that workers could earn $6 per hour and were available to workers with a 95% approval rating and who lived in the United States. Each task took approximately 4 minutes to complete and were given 2.5 hours to complete the task to ensure workers were not pressured for time.

We placed 3 attention checks in each task. Attention checks are gold-standard verification questions whose answers were already known. Attention checks were randomly placed within the series of 58 verifications to gauge how well a worker performed on the given task. To avoid workers from incorrectly marking an attention check due to subjective interpretation of the description, we manually marked these attention checks correct or incorrect. Examples of attention checks are shown in Figure 5.9. Incorrect attention checks were completely mismatched from their image; for example “A very tall sailboat” was used as an incorrect attention check matched to an image of a lady wearing a white dress. We created a total of 11,290 unique attention checks to prevent workers from simply memorizing the attention checks.

Even though these attention checks were designed to be obviously correct or incorrect, we ensured that we do not reject a worker’s submission based off a single, careless mistake or an unexpected ambiguous attention check. After completing a task, each worker’s submission is immediately accepted or rejected based on a rating, which is calculated as the percentage of the last 30 attention checks correctly labeled. If a worker’s rating falls below the threshold of acceptable quality, their task is rejected. However, to ensure fair payment, even if a worker’s rating is below the threshold, their task is accepted if they get all the attention checks in the current task correct. This enables workers who are below the threshold to perform carefully and improve their rating as they continue to do more tasks.
5.4.2 Experiment Setup

Our goal is to vary the acceptance threshold to see how it impacts worker quality over time. We performed a between-subjects $2 \times 2$ study where we varied threshold and transparency. We ran an initial study with a different set of 100 workers to estimate how people performed on this verification task. We found that workers get a mean accuracy of $94 \pm 10\%$ with a median accuracy of $95.5\%$. We chose the thresholds such that the high threshold condition asked workers to perform above the median and the low threshold was below $2 \times$ the standard deviation, allowing workers plenty of room to make mistakes. The high threshold factor level was set at 96% while the low threshold factor level was set at 70%. Workers in the high threshold level could only incorrectly label at most 1 out of 30 of the previous attention checks to avoid rejection, while workers in the low threshold level could error on 8 out of the past 30 attention checks.

We used two levels of transparency: high and low. In the high factor level, workers were able to see their current rating at the beginning of every task and were also alerted of how their rating changed after submitting each task. Meanwhile, in the low factor level, workers did not see their rating, nor did they know what their assigned threshold was.

We recruited workers from AMT for the study and randomized them between conditions. We measured workers’ accuracy and total number of completed tasks under these four conditions.
5.4.3 Data Collected

By the end of the study, 1,134 workers completed 11,666 tasks. In total, 676,628 binary verification questions were answered, of which 34,998 were attention checks. Table 5.2 shows the breakdown of the number of workers who completed at least 1 task. Not all workers who accepted tasks completed them. In the high threshold condition, 106 and 116 workers did not complete any tasks in the high and low transparency conditions respectively. Similarly, 137 and 138 workers did not complete tasks in the low threshold. This resulted in 29 and 28 more people in the low threshold that completed tasks. Workers completed on average a total of 576 verifications each.

5.4.4 Results

On average, the accuracy of the work submitted by workers in all four conditions remained consistent (Figure 5.10). In the low threshold factor level, workers averaged a rating of 93.6±4.7% and 93.3±5.8 in the high and low transparency factor levels. Meanwhile, when the threshold was high, workers in the low transparency factor level averaged 94.2±4.4% while the workers in the high transparency factor level averaged 95.2±4.0%. Overall, the high transparency factor level had a smaller standard deviation throughout the course of workers’ lifetimes. We conducted a two-way ANOVA using the two factors as independent variables on all workers who performed more than 5 tasks. The ANOVA found that there was no significant effect of threshold (F(1, 665)=0.55, p=0.45) or transparency (F(1, 665)=2.29, p=0.13), and no interaction effect (F(1, 665)=0.24, p=0.62). Thus, worker accuracy was unaffected by the accuracy requirement of the task.

Unlike accuracy, worker retention was influenced by our manipulation. By the 50th task, less
Table 5.2: Data collected from the verification experiment. A total of 1,134 workers were divided up into four conditions, with a high or low threshold and transparency.

than 10% of the initial worker population continued to complete tasks. This result is consistent with our observations with the Visual Genome datasets and from previous literature that explains that a small percentage of workers complete most of the crowdsourced work [141]. We also observe that workers in the high threshold and high transparency condition have a sharper dropout rate in the beginning. To measure the effects of the four conditions on dropout, we analyzed the logarithm of the number of tasks completed per condition using an ANOVA. (Log-transforming the data ensured that it was normally distributed and thus amenable to ANOVA analysis.) The ANOVA found that there was a significant effect of transparency ($F(1, 665)=279.87, p<0.001$) and threshold ($F(1, 665)=88.61, p<0.001$), and also a significant interaction effect ($F(1, 665)=76.23, p<0.001$). A post hoc Tukey test [239] showed that the (1) high transparency and high threshold condition had significantly less retention than the (2) low transparency and high threshold condition ($p<0.05$).

5.4.5 Discussion

Workers are consistent in their quality level. With this experiment, we are now ready to answer whether workers are consistent or satisficing to an acceptance threshold. Given that workers’ quality was consistent throughout all the four conditions, evidence suggests that workers were consistent, regardless of the threshold at which requesters accept their work. In the low threshold and high transparency condition, workers are aware that their work will be accepted if their rating is above 70%, and still perform with an average rating of 94%. Workers are risk-averse, and seek to avoid harms to their acceptance rate [156]. Once they find a strategy that allows their work to be accepted, they stick to that strategy throughout their lifetime [162]. This result is consistent with the earlier observational data analysis.

Workers minimize risk by opting out of tasks above their natural accuracy level. If workers do not adjust their quality level in response to task difficulty, the only other possibility is that workers self-select out of tasks they cannot complete effectively. Our data supports this hypothesis: workers in the high transparency and high threshold condition did statistically fewer tasks on average. The workers self-selected out of the task when they had a higher chance of rejection. Out of 267 workers in the high transparency and high threshold condition, 200 workers stopped working once their rating dropped below the 96% threshold. Meanwhile, in the high transparency and low threshold conditions.
CHAPTER 5. LONG-TERM CROWD WORKER QUALITY

(а) Threshold: low (70) and transparency: low, accuracy: 93.3 ± 5.8%  
(b) Threshold: low (70) and transparency: high, accuracy: 93.6 ± 4.7%  
(c) Threshold: high (96) and transparency: low, accuracy: 94.2 ± 4.4%  
(d) Threshold: high (96) and transparency: high, accuracy: 95.2 ± 4.0%

Figure 5.10: Worker accuracy was unaffected by the threshold level and by the visibility of the threshold. The dotted black line indicates the threshold that the workers were supposed to adhere to.

condition, out of the 300 workers who completed our tasks, almost all of them continued working even if their rating dropped below the 70% threshold, often bringing their rating back up to above 96%.

This study illustrates that workers are consistent over very long periods of hundreds of tasks. They quickly develop a strategy to complete the task within the first few tasks and stick with it throughout their lifetime. If their work is approved, they continue to complete the task using the same strategy. If their strategy begins to fail, instead of adapting, they self-select themselves out of the task.

5.5 Predicting From Small Glimpses

The longitudinal analysis in the first section and the experimental analysis in the second section found that crowd worker quality remains consistent regardless of how many tasks the worker completes and regardless of the required acceptance criteria. Bolstered by this result, this section demonstrates the efficacy of predicting a worker’s future quality by observing a small glimpse of their initial work. The ability to predict a worker’s quality on future tasks can help requesters identify good workers and
CHAPTER 5. LONG-TERM CROWD WORKER QUALITY

Figure 5.11: It is possible to model a worker’s future quality by observing only a small glimpse of their initial work. Our *all workers’ average* baseline assumes that all workers perform similarly and manages an error in individual worker quality prediction of 6.9%. Meanwhile, by just observing the first 5 tasks, our *average* and *sigmoid* models achieve 3.4% and 3.7% prediction error respectively. As we observe more hits, the *sigmoid* model is able to represent workers better than the *average* model.

improve the quality of data collected.

5.5.1 Experimental Setup

To create a prediction model, we use the question answering dataset. Our aim is to predict a worker’s quality on the task towards the end of their lifetime. Since workers’ individual quality on every single task can be noisy, we estimate a worker’s future quality as the average of their accuracy on the last 10% of their tasks in their lifetime.

We allow our model to use between the first 5 and the first 200 tasks completed by a worker to estimate their future quality. Therefore, we only test our model on workers who have completed at least 200 tasks. As a baseline, we calculate the *average of all workers’* performances on their last $n$ tasks. We use this value as our guess for each individual worker’s future quality. This model assumes a worker does as well as the average worker does on their final tasks.

Besides the baseline, we use two separate models to estimate a worker’s future quality: *average* and *sigmoid* models. The average model is a simple model that uses the average of the worker’s $n$ tasks as the estimate for all future quality predictions. For example, if a worker averages 90% accuracy on their first five tasks, the average model would predict that the worker will continue to perform at a 90% accuracy. However, if the worker’s quality on their last 10% of tasks is 85%, then the prediction error would be 5%. The *sigmoid* model attempts to represent a worker’s quality as a sigmoid curve with 4 parameters to adjust for the offset of the curve. We use a sigmoid model because
we find that many workers display a very brief learning curve over the first few tasks and remain consistent thereafter. The initial adjustment and future consistency closely resembles a sigmoid curve.

5.5.2 Results

The average of all workers’ accuracy is 87.8%. Using this value as a baseline model for quality yields an error of 6.9%. We plot the error of the baseline as a dotted line in Figure 5.11. The average model performs better: even for only a glimpse of \( n = 5 \) tasks, its error is 3.4%. After seeing a worker’s first \( n = 200 \) tasks, the model gets slightly better and has a prediction error of 3.0%. The sigmoid model outperforms the baseline but underperforms the average model and achieves an error of 3.7% for \( n = 5 \). As the model incorporates more tasks, it becomes the most accurate, managing an error rate of 1.4% after seeing \( n = 200 \) tasks. Furthermore, the model’s standard deviation of the error also decreases from 3.4% to 0.7% as \( n \) increases.

5.5.3 Discussion

Even a glimpse of five tasks can predict a worker’s future quality. Since workers are consistent over time, both the average and the sigmoid models are able to model workers’ quality with very little error. When workers initially start doing work, a simple average model is a good choice for a model to estimate how well the worker might perform in the future. However, as the worker completes more and more tasks, the sigmoid model is able to capture the initial adjustment a worker makes when starting a task. By utilizing such models, requesters can estimate which workers are most likely to produce good work and can easily qualify good workers for long-term work.

5.6 Implications for Crowdsourcing

Encouraging diversity. The consistent accuracy and constant diversity of worker output over time makes sense from a practical perspective: workers are often acclimating to a certain style of completing work [156] and often adopt a particular strategy to get paid. However, this formulaic approach might run counter to a requester’s desire to have richly diverse responses. Checks to increase diversity, such as enforcing a high threshold for diversity, should be employed without fear of worker quality as we have observed that quality does not significantly change with varying acceptance thresholds. Therefore, designing tasks that promote diversity without effecting the annotation quality is a ripe area for future research.

Worker retention. Additional experience affects completion speeds but does not translate to higher quality data. Much work has been done to retain workers [39, 46, 133], but, as shown, retention does not equate to increases in worker quality — just more work completed. Further work
should be conducted to not only retain a worker pool, but also examine methods of identifying good workers and more direct interventions for training poorly performing workers.

Additionally, other studies have shown that the motivation of workers is the predominant factor in the development of fatigue, rather than the total time worked. Although crowdsourcing can be intrinsically motivated, the microtask paradigm found in the majority of crowdsourcing tasks favors a structure that is efficient for workers rather than being interesting for them. Future tasks should consider building continuity in their workflow design for both individual worker efficiency and overall throughput and retention.

Person-centric versus process-centric crowdsourcing. Attaining high quality judgments from crowd workers is often seen as a challenge. This challenge has catalyzed studies suggesting quality control measures that address the problem of noisy or low quality work. While we explored process-centric measures like varying the acceptance or transparency threshold, previous work has experimented with varying financial incentives. All the results support the conclusion that process-centric strategies do not have significant difference in the quality of work submitted. While we agree that such process focused strategies are important to explore, our data reinforces that person-centric strategies (like utilizing worker approval ratings or worker quality on initial tasks) may be more effective because they identify a worker’s (consistent) quality early on.

Limitations. Our analysis solely focuses on data labeling microtasks, and we have not yet studied whether our findings translate over to more complex tasks, such as designing an advertisement or editing an essay. Furthermore, we focus on weeks-to-months crowd worker behavior based on datasets collected over a few months, but there exist some crowdsourcing tasks that have persisted far longer than our study. Thus, we leave the analysis of crowd worker behavior spanning multiple years to future work.

5.7 Conclusion

Microtask crowdsourcing is rapidly being adopted to generate large datasets with millions of labels. Under the Pareto principle, a small minority of workers complete a great majority of the work. In this paper, we studied how the quality of workers’ submissions change over extended periods of time as they complete thousands of tasks. Contrary to previous literature on fatigue and satisficing, we found that workers are extremely consistent throughout their lifetime of submitting work. They adopt a particular strategy for completing tasks and continue to use that strategy without change. To understand how workers settle upon their strategy, we conducted an experiment where we vary the required quality for large crowdsourcing tasks. We found that workers do not satisfice and consistently perform at their usual quality level. If their natural quality level is below the acceptance threshold,
workers tend to opt out from completing further tasks. Due to this consistency, we demonstrated that brief glimpses of just the first five tasks can predict a worker's long-term quality. We argue that such consistent worker behavior must be utilized to develop new crowdsourcing strategies that find good workers and collect unvarying high quality annotations.
Chapter 6

Leveraging Representations in Visual Genome

6.1 Introduction

Thus far, we have presented the Visual Genome dataset, improved its crowdsourcing pipeline, and analyzed the types of annotations included. With such rich information provided, numerous perceptual and cognitive tasks can be tackled. In this section, we aim to provide baseline experimental results using components of Visual Genome that have not been extensively studied. Object detection is already a well-studied problem [54, 71, 212, 70, 190]. Similarly, region graphs and scene graphs have been shown to improve semantic image retrieval [102, 210]. We therefore focus on the remaining components, i.e. attributes, relationships, region descriptions, and question answer pairs.

In Section 6.1.1 we present results for two experiments on attribute prediction. In the first, we treat attributes independently from objects and train a classifier for each attribute, i.e. a classifier for \textit{red} or a classifier for \textit{old}, as in [148, 211, 61, 57, 102]. In the second experiment, we learn object and attribute classifiers \textit{jointly} and predict object-attribute pairs (e.g. predicting that an apple is red), as in [204].

In Section 6.1.2 we present two experiments on relationship prediction. In the first, we aim to predict the predicate between two objects, e.g. predicting the predicate \textit{kicking} or \textit{wearing} between two objects. This experiment is synonymous with existing work in action recognition [79, 185]. In another experiment, we study relationships by classifying jointly the objects and the predicate (e.g. predicting \textit{kicking}(\textit{man, ball})); we show that this is a very difficult task due to the high variability in the appearance of a relationship (e.g. the \textit{ball} might be on the ground or in mid-air above the \textit{man}). These experiments are generalizations of tasks that study spatial relationships between objects and ones that jointly reason about the interaction of humans with objects [270, 184].
CHAPTER 6. LEVERAGING REPRESENTATIONS IN VISUAL GENOME

In Section 6.1.3 we present results for region captioning. This task is closely related to image captioning \[30\]; however, results from the two are not directly comparable, as region descriptions are short, incomplete sentences. We train one of the top 16 state-of-the-art image caption generators \[109\] on (1) our dataset to generate region descriptions and on (2) Flickr30K \[275\] to generate sentence descriptions. To compare results between the two training approaches, we use simple templates to convert region descriptions into complete sentences. For a more robust evaluation, we validate the descriptions we generate using human judgment.

Finally, in Section 6.1.4, we experiment on visual question answering, i.e. given an image and a question, we attempt to provide an answer for the question. We report results on the retrieval of the correct answer from a list of existing answers.

### 6.1.1 Attribute Prediction

Attributes are becoming increasingly important in the field of computer vision, as they offer higher-level semantic cues for various problems and lead to a deeper understanding of images. We can express a wide variety of properties through attributes, such as form (sliced), function (decorative), sentiment (angry), and even intention (helping). Distinguishing between similar objects \[97\] leads
to finer-grained classification, while describing a previously unseen class through attributes shared with known classes can enable “zero-shot” learning [57, 126]. Visual Genome is the largest dataset of attributes, with 26 attributes per image for more than 2.8 million attributes.

Setup. For both experiments, we focus on the 100 most common attributes in our dataset. We only use objects that occur at least 100 times and are associated with one of the 100 attributes in at least one image. For both experiments, we follow a similar data pre-processing pipeline. First, we lowercase, lemmatize [10], and strip excess whitespace from all attributes. Since the number of examples per attribute class varies, we randomly sample 500 attributes from each category (if fewer than 500 are in the class, we take all of them). We end up with around 50,000 attribute instances and 43,000 object-attribute pair instances in total. We use 80% of the images for training and 10% each for validation and testing. Because each image has about the same number of examples, this results in an approximately 80%-10%-10% split over the attributes themselves. The input data for this experiment is the cropped bounding box of the object associated with each attribute.

We train an attribute predictor by using features learned from a convolutional neural network. Specifically, we use a 16-layer VGG network [219] pre-trained no ImageNet and fine-tune it for both of these experiments using the 50,000 attribute and 43,000 object-attribute pair instances respectively. We modify the network so that the learning rate of the final fully-connected layer is 10 times that of the other layers, as this improves convergence time. Convergence is measured as the performance on the validation set. We use a base learning rate of 0.001, which we scale by 0.1 every 200 iterations, and momentum and weight decays of 0.9 and 0.0005 respectively. We use the fine-tuned features from the network and train 100 individual SVMs [84] to predict each attribute. We output multiple attributes for each bounding box input. For the second experiment, we also output the object class.

Results. Table 6.1 shows results for both experiments. For the first experiment on attribute prediction, we converge after around 700 iterations with 18.97% top-one accuracy and 43.11% top-five accuracy. Thus, attributes (like objects) are visually distinguishable from each other. For the second experiment where we also predict the object class, we converge after around 400 iterations with 43.17% top-one accuracy and 71.97% top-five accuracy. Predicting objects jointly with attributes increases the top-one accuracy from 18.97% to 43.17%. This implies that some attributes occur exclusively with a small number of objects. Additionally, by jointly learning attributes with objects, we increase the inter-class variance, making the classification process an easier task.

Figure 6.1 (a) shows example predictions for the first attribute prediction experiment. In general, the model is good at associating objects with their most salient attributes, for example, animal with stuffed and elephant with grazing. However, the crowdsourced ground truth answers sometimes do not contain all valid attributes, so the model is incorrectly penalized for some accurate/true predictions. For example, the white stuffed animal is correct but evaluated as
Table 6.1: (First row) Results for the attribute prediction task where we only predict attributes for a given image crop. (Second row) Attribute-object prediction experiment where we predict both the attributes as well as the object from a given crop of the image.

The attribute clique graphs in Section 2.4.4 clearly show that learning attributes can help us identify types of objects. This experiment strengthens that insight. We learn that studying attributes together with objects can improve attribute prediction.

6.1.2 Relationship Prediction

While objects are the core building blocks of an image, relationships put them in context. These relationships help distinguish between images that contain the same objects but have different holistic interpretations. For example, an image of “a man riding a bike” and “a man falling off a bike” both contain man and bike, but the relationship (riding vs. falling off) changes how we perceive both situations. Visual Genome is the largest known dataset of relationships, with more than 2.3 million relationships and an average of 21 relationships per image.

Setup. The setups of both experiments are similar to those of the experiments we performed on attributes. We again focus on the top 100 most frequent relationships. We lowercase, lemmatize [10], and strip excess whitespace from all relationships. We end up with around 34,000 unique relationship types and 27,000 unique subject-relationship-object triples for training, validation, and testing. The input data to the experiment is the image region containing the union of the bounding boxes of the subject and object (essentially, the bounding box containing the two object boxes). We fine-tune a 16-layer VGG network [219] with the same learning rates mentioned in Section 6.1.1.

Results. Overall, we find that relationships are only slightly visually distinct enough for our discriminative model to learn effectively. Table 6.2 shows results for both experiments. For relationship classification, we converge after around 800 iterations with 8.74% top-one accuracy and 29.69%
Figure 6.2: (a) Example predictions from the relationship prediction experiment. Relationships in the first row are predicted correctly, those in the second row differ from the ground truth but still correctly classify a relationship in the image, and those in the third row are classified incorrectly. The model learns to associate animals leaning towards the ground as eating or drinking and bikes with riding. (b) Example predictions from the relationship-objects prediction experiment. The figure is organized in the same way as Figure (a). The model is able to predict the salient features of the image but fails to distinguish between different objects (e.g. boy and woman and car and bus in the bottom row).

top-five accuracy. Unlike attribute prediction, the accuracy results for relationships are much lower because of the high intra-class variability of most relationships. For the second experiment jointly predicting the relationship and its two object classes, we converge after around 450 iterations with 25.83% top-one accuracy and 65.57% top-five accuracy. We notice that object classification aids relationship prediction. Some relationships occur with some objects and never others; for example, the relationship drive only occurs with the object person and never with any other objects (dog, chair, etc.).

Figure 6.2 (a) shows example predictions for the relationship classification experiment. In general, the model associates object categories with certain relationships (e.g. animals with eating or drinking, bikes with riding, and kids with playing).

Figure 6.2 (b), structured as in Figure 6.2 (a), shows example predictions for the joint prediction of relationships with its objects. The model is able to predict the salient features of the image (e.g. “boat in water”) but fails to distinguish between different objects (e.g. boy vs. woman and car vs. bus in the bottom row).
CHAPTER 6. LEVERAGING REPRESENTATIONS IN VISUAL GENOME

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Top-1 Accuracy</th>
<th>Top-5 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub./Rel./Obj.</td>
<td>25.83%</td>
<td>65.57%</td>
</tr>
</tbody>
</table>

Table 6.2: Results for relationship classification (first row) and joint classification (second row) experiments.

6.1.3 Generating Region Descriptions

Generating sentence descriptions of images has gained popularity as a task in computer vision [111, 150, 109, 247]; however, current state-of-the-art models fail to describe all the different events captured in an image and instead provide only a high-level summary of the image. In this section, we test how well state-of-the-art models can caption the details of images. For both experiments, we use the NeuralTalk model [109], since it not only provides state-of-the-art results but also is shown to be robust enough for predicting short descriptions. We train NeuralTalk on the Visual Genome dataset for region descriptions and on Flickr30K [275] for full sentence descriptions. As a model trained on other datasets would generate complete sentences and would not be comparable [30] to our region descriptions, we convert all region descriptions generated by our model into complete sentences using predefined templates [91].

Figure 6.3: Example predictions from the region description generation experiment by a model trained on Visual Genome region descriptions. Regions in the first column (left) accurately describe the region, and those in the second column (right) are incorrect and unrelated to the corresponding region.
CHAPTER 6. LEVERAGING REPRESENTATIONS IN VISUAL GENOME

Setup. For training, we begin by preprocessing region descriptions; we remove all non-alphameric characters and lowercase and strip excess whitespace from them. We have 5,406,939 region descriptions in total. We end up with 3,784,857 region descriptions for training – 811,040 each for validation and testing. Note that we ensure descriptions of regions from the same image are exclusively in the training, validation, or testing set. We feed the bounding boxes of the regions through the pretrained VGG 16-layer network [219] to get the 4096-dimensional feature vectors of each region. We then use the NeuralTalk [109] model to train a long short-term memory (LSTM) network [89] to generate descriptions of regions. We use a learning rate of 0.001 trained with rmsprop [42]. The model converges after four days.

For testing, we crop the ground-truth region bounding boxes of images and extract their 4096-dimensional 16-layer VGG network [219] features. We then feed these vectors through the pretrained NeuralTalk model to get predictions for region descriptions.

Results. Table 6.3 shows the results for the experiment. We calculate BLEU [175], CIDEr [242], and METEOR [45] scores [30] between the generated descriptions and their ground-truth descriptions. In all cases, the model trained on VisualGenome performs better. Moreover, we asked crowd workers to evaluate whether a generated description was correct—we got 1.6% and 43.03% for models trained on Flickr30K and on Visual Genome, respectively. The large increase in accuracy when the model trained on our data is due to the specificity of our dataset. Our region descriptions are shorter and cover a smaller image area. In comparison, the Flickr30K data are generic descriptions of entire images with multiple events happening in different regions of the image. The model trained on our data is able to make predictions that are more likely to concentrate on the specific part of the image it is looking at, instead of generating a summary description. The objectively low accuracy in both cases illustrates that current models are unable to reason about complex images.

Figure 6.3 shows examples of regions and their predicted descriptions. Since many examples have short descriptions, the predicted descriptions are also short as expected; however, this causes the model to fail to produce more descriptive phrases for regions with multiple objects or with distinctive objects (i.e. objects with many attributes). While we use templates to convert region descriptions into sentences, future work can explore smarter approaches to combine region descriptions and generate a paragraph connecting all the regions into one coherent description.

6.1.4 Question Answering

Visual Genome is currently the largest dataset of visual question answers with more than 1.7 million question and answer pairs. Each of our 108,077 images contains an average of 17 question answer pairs. Answering questions requires a deeper understanding of an image than generic image captioning, Question answering can involve fine-grained recognition (e.g. “What is the breed of the dog?”), object detection (e.g. “Where is the kite in the image?”), activity recognition (e.g. “What is this
Table 6.3: Results for the region description generation experiment. Scores in the first row are for the region descriptions generated from the NeuralTalk model trained on Flickr8K, and those in the second row are for those generated by the model trained on Visual Genome data. BLEU, CIDEr, and METEOR scores all compare the predicted description to a ground truth in different ways.

<table>
<thead>
<tr>
<th></th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
<th>CIDEr</th>
<th>METEOR</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flickr8K</td>
<td>0.09</td>
<td>0.01</td>
<td>0.002</td>
<td>0.0004</td>
<td>0.05</td>
<td>0.04</td>
<td>1.6%</td>
</tr>
<tr>
<td>VG</td>
<td>0.17</td>
<td>0.05</td>
<td>0.02</td>
<td>0.01</td>
<td>0.30</td>
<td>0.09</td>
<td>43.03%</td>
</tr>
</tbody>
</table>

Table 6.4: Baseline QA performances in the 6 different question types. We report human evaluation as well as a baseline method that predicts the most frequently occurring answer in the dataset.

<table>
<thead>
<tr>
<th></th>
<th>top-100</th>
<th>top-500</th>
<th>top-1000</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td>What</td>
<td>0.420</td>
<td>0.602</td>
<td>0.672</td>
<td>0.965</td>
</tr>
<tr>
<td>Where</td>
<td>0.096</td>
<td>0.324</td>
<td>0.418</td>
<td>0.957</td>
</tr>
<tr>
<td>When</td>
<td>0.714</td>
<td>0.809</td>
<td>0.834</td>
<td>0.944</td>
</tr>
<tr>
<td>Who</td>
<td>0.355</td>
<td>0.493</td>
<td>0.605</td>
<td>0.965</td>
</tr>
<tr>
<td>Why</td>
<td>0.034</td>
<td>0.118</td>
<td>0.187</td>
<td>0.927</td>
</tr>
<tr>
<td>How</td>
<td>0.780</td>
<td>0.827</td>
<td>0.846</td>
<td>0.942</td>
</tr>
<tr>
<td>Overall</td>
<td>0.411</td>
<td>0.573</td>
<td>0.641</td>
<td>0.966</td>
</tr>
</tbody>
</table>

man doing?”), knowledge base reasoning (e.g. “Is this glass full?”), and common-sense reasoning (e.g. “What street will we be on if we turn right?”).

By leveraging the detailed annotations in the scene graphs in Visual Genome, we envision building smart models that can answer a myriad of visual questions. While we encourage the construction of smart models, in this paper, we provide some baseline results to help others compare their models.

**Setup.** We split the QA pairs into a training set (60%) and a test set (40%). We ensure that all images are exclusive to either the training set or the test set. We implement a simple baseline model that relies on answer frequency. The model counts the top \( k \) most frequent answers (similar to the ImageNet challenge \[198\]) in the training set as the predictions for all the test questions, where \( k = 100, 500, \) and 1000. We let a model make \( k \) different predictions. We say the model is correct on a QA if one of the \( k \) predictions matches exactly with the ground-truth answer. We report the accuracy over all test questions. This evaluation method works well when the answers are short, especially for single-word answers. However, it causes problems when the answers are long phrases and sentences. We also report humans performance (similar to previous work \[2, 277\]) on these questions by presenting them with the image and the question along with 10 multiple choice answers out of which one of them was the ground truth and the other 9 were randomly chosen from the dataset. Other evaluation methods require word ontologies \[146\].
Results. Table 6.4 shows the performance of the open-ended visual question answering task. These baseline results imply the long-tail distribution of the answers. Long-tail distribution is common in existing QA datasets as well [2, 146]. The top 100, 500, and 1000 most frequent answers only cover 41.1%, 57.3%, and 64.1% of the correct answers. In comparison, the corresponding sets of frequent answers in VQA [2] cover 63%, 75%, and 80% of the test set answers. The “where” and “why” questions, which tend to involve spatial and common sense reasoning, tend to have more diverse answers and hence perform poorly, with performances of 9.36% and 3.4% top-100 respectively. The top 1000 frequent answers cover only 41.8% and 18.7% of the correct answers from these two question types respectively. In comparison, humans perform extremely well in all the questions types achieving an overall accuracy of 96.6%.
Chapter 7

Dense-Captioning Events in Videos

7.1 Introduction

With the introduction of large scale activity datasets, it has become possible to categorize videos into a discrete set of action categories. For example, in Figure 7.1, such models would output labels like playing piano or dancing. While the success of these methods is encouraging, they all share one key limitation: detail. To elevate the lack of detail from existing action detection models, subsequent work has explored explaining video semantics using sentence descriptions. For example, in Figure 7.1, such models would likely concentrate on an elderly man playing the piano in front of a crowd. While this caption provides us more details about who is playing the piano and mentions an audience, it fails to recognize and articulate all the other events in the video. For example, at some point in the video, a woman starts singing along with the pianist and then later another man starts dancing to the music. In order to identify all the events in a video and describe them in natural language, we introduce the task of dense-captioning events, which requires a model to generate a set of descriptions for multiple events occurring in the video and localize them in time.

Dense-captioning events is analogous to dense-image-captioning: it describes videos and localize events in time whereas dense-image-captioning describes and localizes regions in space. However, we observe that dense-captioning events comes with its own set of challenges distinct from the image case. One observation is that events in videos can range across multiple time scales and can even overlap. While piano recitals might last for the entire duration of a long video, the applause takes place in a couple of seconds. To capture all such events, we need to design ways of encoding short as well as long sequences of video frames to propose events. Past captioning works

My main contributions to dense-video captioning involved helping build components of the captioning model (specifically the temporal localization), benchmarking our results against previous work, and gathering statistics about the Activity Net Captions dataset.
An elderly man is playing the piano in front of a crowd. Another man starts dancing to the music, gathering attention from the crowd. Eventually the elderly man finishes playing and hugs the woman, and the crowd applaud.

A woman walks to the piano and briefly talks to the elderly man. The woman starts singing along with the pianist. Another man starts dancing to the music, gathering attention from the crowd. Eventually the elderly man finishes playing and hugs the woman, and the crowd applaud.

Figure 7.1: Dense-captioning events in a video involves detecting multiple events that occur in a video and describing each event using natural language. These events are temporally localized in the video with independent start and end times, resulting in some events that might also occur concurrently and overlap in time.

We have circumvented this problem by encoding the entire video sequence by mean-pooling or by using a recurrent neural network (RNN). While this works well for short clips, encoding long video sequences that span minutes leads to vanishing gradients, preventing successful training. To overcome this limitation, we extend recent work on generating action proposals to multi-scale detection of events. Also, our proposal module processes each video in a forward pass, allowing us to detect events as they occur.

Another key observation is that the events in a given video are usually related to one another. In Figure 7.1 the crowd applauds because a man was playing the piano. Therefore, our model must be able to use context from surrounding events to caption each event. A recent paper has attempted to describe videos with multiple sentences. However, their model generates sentences for instructional “cooking” videos where the events occur sequentially and highly correlated to the objects in the video. We show that their model does not generalize to “open” domain videos where events are action oriented and can even overlap. We introduce a captioning module that utilizes the context from all the events from our proposal module to generate each sentence. In addition, we show a variant of our captioning module that can operate on streaming videos by attending over only the past events. Our full model attends over both past as well as future events.
and demonstrates the importance of using context.

To evaluate our model and benchmark progress in dense-captioning events, we introduce the ActivityNet Captions dataset. ActivityNet Captions contains 20k videos taken from ActivityNet, where each video is annotated with a series of temporally localized descriptions (Figure 7.1). To showcase long term event detection, our dataset contains videos as long as 10 minutes, with each video annotated with on average 3.65 sentences. The descriptions refer to events that might be simultaneously occurring, causing the video segments to overlap. We ensure that each description in a given video is unique and refers to only one segment. While our videos are centered around human activities, the descriptions may also refer to non-human events such as: two hours later, the mixture becomes a delicious cake to eat. We collect our descriptions using crowdsourcing find that there is high agreement in the temporal event segments, which is in line with research suggesting that brain activity is naturally structured into semantically meaningful events.

With ActivityNet Captions, we are able to provide the first results for the task of dense-captioning events. Together with our online proposal module and our online captioning module, we show that we can detect and describe events in long or even streaming videos. We demonstrate that we are able to detect events found in short clips as well as in long video sequences. Furthermore, we show that utilizing context from other events in the video improves dense-captioning events. Finally, we demonstrate how ActivityNet Captions can be used to study video retrieval as well as event localization.

### 7.2 Related work

Dense-captioning events bridges two separate bodies of work: temporal action proposals and video captioning. First, we review related work on action recognition, action detection and temporal proposals. Next, we survey how video captioning started from video retrieval and video summarization, leading to single-sentence captioning work. Finally, we contrast our work with recent work in captioning images and videos with multiple sentences.

Early work in activity recognition involved using hidden Markov models to learn latent action states, followed by discriminative SVM models that used key poses and action grammars. Similar works have used hand-crafted features or object-centric features to recognize actions in fixed camera settings. More recent works have used dense trajectories or deep learning features to study actions. While our work is similar to these methods, we focus on describing such events with natural language instead of a fixed label set.

To enable action localization, temporal action proposal methods started from traditional sliding window approaches and later started building models to propose a handful of possible action segments. These proposal methods have used dictionary learning or RNN

---

1 The dataset is available at [http://cs.stanford.edu/people/ranjaykrishna/densevid/](http://cs.stanford.edu/people/ranjaykrishna/densevid/) For a detailed analysis of our dataset, please see our supplementary material.
architectures to find possible segments of interest. However, such methods required each video frame to be processed once for every sliding window. DAPs introduced a framework to allow proposing overlapping segments using a sliding window. We modify this framework by removing the sliding windows and outputting proposals at every time step in a single pass of the video. We further extend this model and enable it to detect long events by implementing a multi-scale version of DAPs, where we sample frames at longer strides.

Orthogonal to work studying proposals, early approaches that connected video with language studied the task of video retrieval with natural language. They worked on generating a common embedding space between language and videos. Similar to these, we evaluate how well existing models perform on our dataset. Additionally, we introduce the task of localizing a given sentence given a video frame, allowing us to now also evaluate whether our models are able to locate specified events.

In an effort to start describing videos, methods in video summarization aimed to congregate segments of videos that include important or interesting visual information. These methods attempted to use low level features such as color and motion or attempted to model objects and their relationships to select key segments. Meanwhile, others have utilized text inputs from user studies to guide the selection process. While these summaries provide a means of finding important segments, these methods are limited by small vocabularies and do not evaluate how well we can explain visual events.

After these summarization works, early attempts at video captioning simply mean-pooled video frame features and used a pipeline inspired by the success of image captioning. However, this approach only works for short video clips with only one major event. To avoid this issue, others have proposed either a recurrent encoder or an attention mechanism. To capture more detail in videos, a new paper has recommended describing videos with paragraphs (a list of sentences) using a hierarchical RNN where the top level network generates a series of hidden vectors that are used to initialize low level RNNs that generate each individual sentence. While our paper is most similar to this work, we address two important missing factors. First, the sentences that their model generates refer to different events in the video but are not localized in time. Second, they use the TACoS-MultiLevel, which contains less than 200 videos and is constrained to “cooking” videos and only contain non-overlapping sequential events. We address these issues by introducing the ActivityNet Captions dataset which contains overlapping events and by introducing our captioning module that uses temporal context to capture the interdependency between all the events in a video.

Finally, we build upon the recent work on dense-image-captioning, which generates a set of localized descriptions for an image. Further work for this task has used spatial context to improve captioning. Inspired by this work, and by recent literature on using spatial attention to improve human tracking, we design our captioning module to incorporate temporal context.
CHAPTER 7. DENSE-CAPTIONS EVENTS IN VIDEOS

7.3 Dense-captioning events model

Overview. Our goal is to design an architecture that jointly localizes temporal proposals of interest and then describes each with natural language. The two main challenges we face are to develop a method that can (1) detect multiple events in short as well as long video sequences and (2) utilize the context from past, concurrent and future events to generate descriptions of each one. Our proposed architecture (Figure 7.2) draws on architectural elements present in recent work on action proposal [53] and social human tracking [1] to tackle both these challenges.

Formally, the input to our system is a sequence of video frames $v = \{v_t\}$ where $t \in 0, ..., T - 1$ indexes the frames in temporal order. Our output is a set of sentences $s_i \in S$ where $s_i = (t_{\text{start}}, t_{\text{end}}, \{v_j\})$ consists of the start and end times for each sentence which is defined by a set of words $v_j \in V$ with differing lengths for each sentence and $V$ is our vocabulary set.

Our model first sends the video frames through a proposal module that generates a set of proposals:

$$P = \{(t_{\text{start}}^i, t_{\text{end}}^i, \text{score}_i, h_i)\} \quad (7.1)$$

All the proposals with a $\text{score}_i$ higher than a threshold are forwarded to our language model that uses context from the other proposals while captioning each event. The hidden representation $h_i$ of the
event proposal module is used as inputs to the captioning module, which then outputs descriptions for each event, while utilizing the context from the other events.

7.3.1 Event proposal module

The proposal module in Figure 7.2 tackles the challenge of detecting events in short as well as long video sequences, while preventing the dense application of our language model over sliding windows during inference. Prior work usually pools video features globally into a fixed sized vector \([49, 244, 262]\), which is sufficient for representing short video clips but is unable to detect multiple events in long videos. Additionally, we would like to detect events in a single pass of the video so that the gains over a simple temporal sliding window are significant. To tackle this challenge, we design an event proposal module to be a variant of DAPs \([53]\) that can detect longer events.

**Input.** Our proposal module receives a series of features capturing semantic information from the video frames. Concretely, the input to our proposal module is a sequence of features: \(f_t = F(v_t : v_{t+\delta})\) where \(\delta\) is the time resolution of each feature \(f_t\). In our paper, \(F\) extracts C3D features \([100]\) where \(\delta = 16\) frames. The output of \(F\) is a tensor of size \(N \times D\) where \(D = 500\) dimensional features and \(N = T/\delta\) discretizes the video frames.

**DAPs.** Next, we feed these features into a variant of DAPs \([53]\) where we sample the videos features at different strides (1, 2, 4 and 8 for our experiments) and feed them into a proposal long short-term memory (LSTM) unit. The longer strides are able to capture longer events. The LSTM accumulates evidence across time as the video features progress. We do not modify the training of DAPs and only change the model at inference time by outputting \(K\) proposals at every time step, each proposing an event with offsets. So, the LSTM is capable of generating proposals at different overlapping time intervals and we only need to iterate over the video once, since all the strides can be computed in parallel. Whenever the proposal LSTM detects an event, we use the hidden state of the LSTM at that time step as a feature representation of the visual event. Note that the proposal model can output proposals for events that can be overlapping. While traditional DAPs uses non-maximum suppression to eliminate overlapping outputs, we keep them separately and treat them as individual events.

7.3.2 Captioning module with context

Once we have the event proposals, the next stage of our pipeline is responsible for describing each event. A naive captioning approach could treat each description individually and use a captioning LSTM network to describe each one. However, most events in a video are correlated and can even cause one another. For example, we saw in Figure 7.1 that the man playing the piano caused the other person to start dancing. We also saw that after the man finished playing the piano, the audience applauded. To capture such correlations, we design our captioning module to incorporate the “context” from its neighboring events. Inspired by recent work \([1]\) on human tracking that utilizes spatial
context between neighboring tracks, we develop an analogous model that captures temporal context in videos by grouping together events in time instead of tracks in space.

**Incorporating context.** To capture the context from all other neighboring events, we categorize all events into two buckets relative to a reference event. These two context buckets capture events that have already occurred (past), and events that take place after this event has finished (future). Concurrent events are split into one of the two buckets: past if it end early and future otherwise. For a given video event from the proposal module, with hidden representation $h_i$ and start and end times of $[t_i^{start}, t_i^{end}]$, we calculate the past and future context representations as follows:

$$h_{i}^{\text{past}} = \frac{1}{Z_{\text{past}}} \sum_{j \neq i} \mathbb{1}[t_j^{end} < t_i^{end}] w_j h_j$$  \hspace{1cm} (7.2)$$

$$h_{i}^{\text{future}} = \frac{1}{Z_{\text{future}}} \sum_{j \neq i} \mathbb{1}[t_j^{end} > t_i^{end}] w_j h_j$$  \hspace{1cm} (7.3)$$

where $h_j$ is the hidden representation of the other proposed events in the video. $w_j$ is the weight used to determine how relevant event $j$ is to event $i$. $Z$ is the normalization that is calculated as $Z_{\text{past}} = \sum_{j \neq i} \mathbb{1}[t_j^{end} < t_i^{end}]$. We calculate $w_j$ as follows:

$$a_i = w_a h_i + b_a$$  \hspace{1cm} (7.4)$$

$$w_j = a_i h_j$$  \hspace{1cm} (7.5)$$

where $a_i$ is the attention vector calculated from the learnt weights $w_a$ and bias $b_a$. We use the dot product of $a_i$ and $h_j$ to calculate $w_j$. The concatenation of $(h_{i}^{\text{past}}, h_{i}, h_{i}^{\text{future}})$ is then fed as the input to the captioning LSTM that describes the event. With the help of the context, each LSTM also has knowledge about events that have happened or will happen and can tune its captions accordingly.

**Language modeling.** Each language LSTM is initialized to have 2 layers with 512 dimensional hidden representation. We randomly initialize all the word vector embeddings from a Gaussian with standard deviation of 0.01. We sample predictions from the model using beam search of size 5.

### 7.3.3 Implementation details.

**Loss function.** We use two separate losses to train both our proposal model ($L_{\text{prop}}$) and our captioning model ($L_{\text{cap}}$). Our proposal models predicts confidences ranging between 0 and 1 for varying proposal lengths. We use a weighted cross-entropy term to evaluate each proposal confidence.

We only pass to the language model proposals that have a high IoU with ground truth proposals. Similar to previous work on language modeling [115, 109], we use a cross-entropy loss across all words in every sentence. We normalize the loss by the batch-size and sequence length in the language model. We weight the contribution of the captioning loss with $\lambda_1 = 1.0$ and the proposal loss with $\lambda_2 = 0.1$:

$$L = \lambda_1 L_{\text{cap}} + \lambda_2 L_{\text{prop}}$$  \hspace{1cm} (7.6)$$
Training and optimization. We train our full dense-captioning model by alternating between training the language model and the proposal module every 500 iterations. We first train the captioning module by masking all neighboring events for 10 epochs before adding in the context features. We initialize all weights using a Gaussian with standard deviation of 0.01. We use stochastic gradient descent with momentum 0.9 to train. We use an initial learning rate of $1 \times 10^{-2}$ for the language model and $1 \times 10^{-3}$ for the proposal module. For efficiency, we do not finetune the C3D feature extraction.

Our training batch-size is set to 1. We cap all sentences to be a maximum sentence length of 30 words and implement all our code in PyTorch 0.1.10. One mini-batch runs in approximately 15.84 ms on a Titan X GPU and it takes 2 days for the model to converge.

7.4 ActivityNet Captions dataset

The ActivityNet Captions dataset connects videos to a series of temporally annotated sentences. Each sentence covers an unique segment of the video, describing an event that occurs. These events may occur over very long or short periods of time and are not limited in any capacity, allowing them to co-occur. We will now present an overview of the dataset and also provide a detailed analysis and
with GT proposals

<table>
<thead>
<tr>
<th></th>
<th>B@1</th>
<th>B@2</th>
<th>B@3</th>
<th>B@4</th>
<th>M</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM-YT</td>
<td>244</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2VT</td>
<td>245</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H-RNN</td>
<td>276</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>no context (ours)</td>
<td>20.35</td>
<td>8.99</td>
<td>4.60</td>
<td>2.62</td>
<td>7.85</td>
<td>20.97</td>
</tr>
<tr>
<td>online—attn (ours)</td>
<td>21.92</td>
<td>9.88</td>
<td>5.21</td>
<td>3.06</td>
<td>8.50</td>
<td>22.19</td>
</tr>
<tr>
<td>online (ours)</td>
<td>22.10</td>
<td>10.02</td>
<td>5.66</td>
<td>3.10</td>
<td>8.88</td>
<td>22.94</td>
</tr>
<tr>
<td>full (ours)</td>
<td>26.45</td>
<td>13.48</td>
<td>7.12</td>
<td>3.98</td>
<td>9.46</td>
<td>24.56</td>
</tr>
</tbody>
</table>

with learnt proposals

<table>
<thead>
<tr>
<th></th>
<th>B@1</th>
<th>B@2</th>
<th>B@3</th>
<th>B@4</th>
<th>M</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM-YT</td>
<td>244</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2VT</td>
<td>245</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H-RNN</td>
<td>276</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>no context (ours)</td>
<td>12.23</td>
<td>3.48</td>
<td>2.10</td>
<td>0.88</td>
<td>3.76</td>
<td>12.34</td>
</tr>
<tr>
<td>online—attn (ours)</td>
<td>15.20</td>
<td>5.43</td>
<td>2.52</td>
<td>1.34</td>
<td>4.18</td>
<td>14.20</td>
</tr>
<tr>
<td>online (ours)</td>
<td>17.10</td>
<td>7.34</td>
<td>3.23</td>
<td>1.89</td>
<td>4.38</td>
<td>15.30</td>
</tr>
<tr>
<td>full (ours)</td>
<td>17.95</td>
<td>7.69</td>
<td>3.86</td>
<td>2.20</td>
<td>4.82</td>
<td>17.29</td>
</tr>
</tbody>
</table>

Table 7.1: We report Bleu (B), METEOR (M) and CIDEr (C) captioning scores for the task of dense-captioning events. On the top table, we report performances of just our captioning module with ground truth proposals. On the bottom table, we report the combined performances of our complete model, with proposals predicted from our proposal module. Since prior work has focused only on describing entire videos and not also detecting a series of events, we only compare existing video captioning models using ground truth proposals.

7.4.1 Dataset statistics

On average, each of the 20k videos in ActivityNet Captions contains 3.65 temporally localized sentences, resulting in a total of 100k sentences. We find that the number of sentences per video follows a relatively normal distribution. Furthermore, as the video duration increases, the number of sentences also increases. Each sentence has an average length of 13.48 words, which is also normally distributed.

On average, each sentence describes 36 seconds and 31% of their respective videos. However, the entire paragraph for each video on average describes 94.6% of the entire video, demonstrating that each paragraph annotation still covers all major actions within the video. Furthermore, we found that 10% of the temporal descriptions overlap, showing that the events cover simultaneous events.

Finally, our analysis on the sentences themselves indicate that ActivityNet Captions focuses on verbs and actions. In Figure 7.3, we compare against Visual Genome [118], the image dataset with most number of image descriptions (4.5 million). With the percentage of verbs comprising comparison with other datasets in our supplementary material.
ActivityNet Captions being significantly more, we find that ActivityNet Captions shifts sentence descriptions from being object-centric in images to action-centric in videos. Furthermore, as there exists a greater percentage of pronouns in ActivityNet Captions, we find that the sentence labels will more often refer to entities found in prior sentences.

### 7.4.2 Temporal agreement amongst annotators

To verify that ActivityNet Captions’s captions mark semantically meaningful events [5], we collected two distinct, temporally annotated paragraphs from different workers for each of the 4926 validation and 5044 test videos. Each pair of annotations was then tested to see how well they temporally corresponded to each other. We found that, on average, each sentence description had an $tIoU$ of 70.2% with the maximal overlapping combination of sentences from the other paragraph. Since these results agree with prior work [5], we found that workers generally agree with each other when annotating temporal boundaries of video events.

### 7.5 Experiments

We evaluate our model by detecting multiple events in videos and describing them. We refer to this task as dense-captioning events (Section 7.5.1). We test our model on ActivityNet Captions, which was built specifically for this task.

Next, we provide baseline results on two additional tasks that are possible with our model. The first of these tasks is localization (Section 7.5.2), which tests our proposal model’s capability to adequately localize all the events for a given video. The second task is retrieval (Section 7.5.3), which tests a variant of our model’s ability to recover the correct set of sentences given the video or vice versa. Both these tasks are designed to test the event proposal module (localization) and the captioning module (retrieval) individually.

### 7.5.1 Dense-captioning events

To dense-caption events, our model is given an input video and is tasked with detecting individual events and describing each one with natural language.

**Evaluation metrics.** Inspired by the dense-image-captioning [101] metric, we use a similar metric to measure the joint ability of our model to both localize and caption events. This metric computes the average precision across $tIoU$ thresholds of 0.3, 0.5, 0.7 when captioning the top 1000 proposals. We measure precision of our captions using traditional evaluation metrics: Bleu, METEOR and CIDEr. To isolate the performance of language in the predicted captions without localization, we also use ground truth locations across each test image and evaluate predicted captions.

**Baseline models.** Since all the previous models proposed so far have focused on the task of describing
CHAPTER 7. DENSE-CAPTIONING EVENTS IN VIDEOS

<table>
<thead>
<tr>
<th></th>
<th>B@1</th>
<th>B@2</th>
<th>B@3</th>
<th>B@4</th>
<th>M</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>no context</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st sen.</td>
<td>23.60</td>
<td>12.19</td>
<td>7.11</td>
<td>4.51</td>
<td>9.34</td>
<td><strong>31.56</strong></td>
</tr>
<tr>
<td>2nd sen.</td>
<td>19.74</td>
<td>8.17</td>
<td>3.76</td>
<td>1.87</td>
<td>7.79</td>
<td>19.37</td>
</tr>
<tr>
<td>3rd sen.</td>
<td>18.89</td>
<td>7.51</td>
<td>3.43</td>
<td>1.87</td>
<td>7.31</td>
<td>19.36</td>
</tr>
<tr>
<td><strong>online</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st sen.</td>
<td>24.93</td>
<td>12.38</td>
<td>7.45</td>
<td>4.77</td>
<td>8.10</td>
<td>30.92</td>
</tr>
<tr>
<td>2nd sen.</td>
<td>19.96</td>
<td>8.66</td>
<td>4.01</td>
<td>1.93</td>
<td>7.88</td>
<td>19.17</td>
</tr>
<tr>
<td>3rd sen.</td>
<td>19.22</td>
<td>7.72</td>
<td>3.56</td>
<td>1.89</td>
<td>7.41</td>
<td>19.36</td>
</tr>
<tr>
<td><strong>full</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st sen.</td>
<td><strong>26.33</strong></td>
<td><strong>13.98</strong></td>
<td><strong>8.45</strong></td>
<td><strong>5.52</strong></td>
<td><strong>10.03</strong></td>
<td><strong>29.92</strong></td>
</tr>
<tr>
<td>2nd sen.</td>
<td><strong>21.46</strong></td>
<td><strong>9.06</strong></td>
<td><strong>4.40</strong></td>
<td><strong>2.33</strong></td>
<td><strong>8.28</strong></td>
<td><strong>20.17</strong></td>
</tr>
<tr>
<td>3rd sen.</td>
<td><strong>19.82</strong></td>
<td><strong>7.93</strong></td>
<td><strong>3.63</strong></td>
<td><strong>1.83</strong></td>
<td><strong>7.81</strong></td>
<td><strong>20.01</strong></td>
</tr>
</tbody>
</table>

Table 7.2: We report the effects of context on captioning the 1st, 2nd and 3rd events in a video. We see that performance increases with the addition of past context in the online model and with future context in full model.

By not detecting a series of events, we only compare existing video captioning models using ground truth proposals. Specifically, we compare our work with LSTM-YT [244], S2VT [245] and H-RNN [276]. LSTM-YT pools together video features to describe videos while S2VT encodes a video using an RNN. H-RNN generates paragraphs by using one RNN to caption individual sentences while the second RNN is used to sequentially initialize the hidden state for the next sentence generation. Our model can be though of as a generalization of the H-RNN model as it uses context, not just from the previous sentence but from surrounding events in the video. Additionally, our method treats context, not as features from object detectors but encodes it from unique parts of the proposal module.

**Variants of our model.** Additionally, we compare different variants of our model. Our no context model is our implementation of S2VT. The full model is our complete model described in Section 7.3. The online model is a version of our full model that uses context only from past events and not from future events. This version of our model can be used to caption long streams of video in a single pass. The full−attn and online−attn models use mean pooling instead of attention to concatenate features, i.e. it sets $w_j = 1$ in Equation 7.5.

**Captioning results.** Since all the previous work has focused on captioning complete videos, We find that LSTM-YT performs much worse than other models as it tries to encode long sequences of video by mean pooling their features (Table 7.1). H-RNN performs slightly better but attends over object level features to generate sentence, which causes it to only slightly outperform LSTM-YT since we demonstrated earlier that the captions in our dataset are not object centric but action centric instead. S2VT and our no context model performs better than the previous baselines with a CIDEr score of 20.97 as it uses an RNN to encode the video features. We see an improvement in performance to 22.19 and 22.94 when we incorporate context from past events into our online−attn
and online models. Finally, we also considering events that will happen in the future, we see further improvements to 24.24 and 24.56 for the full—atta and full models. Note that while the improvements
Figure 7.5: Evaluating our proposal module, we find that sampling videos at varying strides does in fact improve the module’s ability to localize events, specially longer events.

```
<table>
<thead>
<tr>
<th></th>
<th>Video retrieval</th>
<th>Paragraph retrieval</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R@1  R@5  R@50 Med.</td>
<td>R@1  R@5  R@50 Med.</td>
</tr>
<tr>
<td>LSTM-YT</td>
<td>244  0.00  0.04  0.24  102</td>
<td>245  0.00  0.07  0.38  98</td>
</tr>
<tr>
<td>no context</td>
<td>245  0.05  0.14  0.32  78</td>
<td>245  0.07  0.18  0.45  56</td>
</tr>
<tr>
<td>online (ours)</td>
<td>0.10  0.32  0.60  36</td>
<td>0.17  0.34  0.70  33</td>
</tr>
<tr>
<td>full (ours)</td>
<td>0.14  0.32  0.65  34</td>
<td>0.18  0.36  0.74  32</td>
</tr>
</tbody>
</table>
```

Table 7.3: Results for video and paragraph retrieval. We see that the utilization of context to encode video events help us improve retrieval. R@k measures the recall at varying thresholds k and med. rank measures the median rank the retrieval.

from using attention is not too large, we see greater improvements amongst videos with more events, suggesting that attention is useful for longer videos.

**Sentence order.** To further benchmark the improvements calculated from utilizing past and future context, we report results using ground truth proposals for the first three sentences in each video (Table 7.2). While there are videos with more than three sentences, we report results only for the first three because almost all the videos in the dataset contains at least three sentences. We notice that the online and full context models see most of their improvements from subsequent sentences, i.e. not the first sentence. In fact, we notice that after adding context, the CIDEr score for the online and full models tend to decrease for the 1st sentence.

**Results for dense-captioning events.** When using proposals instead of ground truth events (Table 7.1), we see a similar trend where adding more context improves captioning. However, we also see that the improvements from attention are more pronounced since there are many events that the model has to caption. Attention allows the model to adequately focus in on select other events that are relevant to the current event. We show examples qualitative results from the variants of our models in Figure 7.4. In (a), we see that the last caption in the no context model drifts off topic
while the full model utilizes context to generate more reasonable context. In (c), we see that our full context model is able to use the knowledge that the vegetables are later mixed in the bowl to also mention the bowl in the third and fourth sentences, propagating context back through to past events. However, context is not always successful at generating better captions. In (c), when the proposed segments have a high overlap, our model fails to distinguish between the two events, causing it to repeat captions.

### 7.5.2 Event localization

One of the main goals of this paper is to develop models that can locate any given event within a video. Therefore, we test how well our model can predict the temporal location of events within the corresponding video, in isolation of the captioning module. Recall that our variant of the proposal module uses proposes videos at different strides. Specifically, we test with strides of 1, 2, 4 and 8. Each stride can be computed in parallel, allowing the proposal to run in a single pass.

**Setup.** We evaluate our proposal module using recall (like previous work [53]) against (1) the number of proposals and (2) the IoU with ground truth events. Specifically, we are testing whether, the use of different strides does in fact improve event localization.

**Results.** Figure 7.5 shows the recall of predicted localizations that overlap with ground truth over a range of IoU’s from 0.0 to 1.0 and number of proposals ranging till 1000. We find that using more strides improves recall across all values of IoU’s with diminishing returns. We also observe that when proposing only a few proposals, the model with stride 1 performs better than any of the multi-stride versions. This occurs because there are more training examples for smaller strides as these models have more video frames to iterate over, allowing them to be more accurate. So, when predicting only a few proposals, the model with stride 1 localizes the most correct events. However, as we increase the number of proposals, we find that the proposal network with only a stride of 1 plateaus around a recall of 0.3, while our multi-scale models perform better.

### 7.5.3 Video and paragraph retrieval

While we introduce dense-captioning events, a new task to study video understanding, we also evaluate our intuition to use context on a more traditional task: video retrieval.

**Setup.** In video retrieval, we are given a set of sentences that describe different parts of a video and are asked to retrieve the correct video from the test set of all videos. Our retrieval model is a slight variant on our dense-captioning model where we encode all the sentences using our captioning module and then combine the context together for each sentence and match each sentence to multiple proposals from a video. We assume that we have ground truth proposals for each video and encode each proposal using the LSTM from our proposal model. We train our model using a max-margin loss that attempts to align the correct sentence encoding to its corresponding video proposal encoding.
We also report how this model performs if the task is reversed, where we are given a video as input and are asked to retrieve the correct paragraph from the complete set of paragraphs in the test set. **Results.** We report our results in Table 7.3. We evaluate retrieval using recall at various thresholds and the median rank. We use the same baseline models as our previous tasks. We find that models that use RNNs (no context) to encode the video proposals perform better than max pooling video features (LSTM-YT). We also see a direct increase in performance when context is used. Unlike dense-captioning, we do not see a marked increase in performance when we include context from future events as well. We find that our online models performs almost at par with our full model.

7.6 Conclusion

We introduced the task of dense-captioning events and identified two challenges: (1) events can occur within a second or last up to minutes, and (2) events in a video are related to one another. To tackle both these challenges, we proposed a model that combines a new variant of an existing proposal module with a new captioning module. The proposal module samples video frames at different strides and gathers evidence to propose events at different time scales in one pass of the video. The captioning module attends over the neighboring events, utilizing their context to improve the generation of captions. We compare variants of our model and demonstrate that context does indeed improve captioning. We further show how the captioning model uses context to improve video retrieval and how our proposal model uses the different strides to improve event localization. Finally, this paper also releases a new dataset for dense-captioning events: ActivityNet Captions.

7.7 Supplementary material

In the supplementary material, we compare and contrast our dataset with other datasets and provide additional details about our dataset. We include screenshots of our collection interface with detailed instructions. We also provide additional details about the workers who completed our tasks.

7.7.1 Comparison to other datasets.

Curation and open distribution is closely correlated with progress in the field of video understanding (Table 7.4). The KTH dataset [208] pioneered the field by studying human actions with a black background. Since then, datasets like UCF101 [226], Sports 1M [110], Thumos 15 [75] have focused on studying actions in sports related internet videos while HMDB 51 [123] and Hollywood 2 [152] introduced a dataset of movie clips. Recently, ActivityNet [22] and Charades [216] broadened the domain of activities captured by these datasets by including a large set of human activities. In an effort to map video semantics with language, MPII MD [193] and M-VAD [237] released short movie clips with descriptions. In an effort to capture longer events, MSR-VTT [263], MSVD [29]...
and YouCook \cite{41} collected a dataset with slightly longer length, at the cost of a few descriptions than previous datasets. To further improve video annotations, KITTI \cite{67} and TACoS \cite{188} also temporally localized their video descriptions. Orthogonally, in an effort to increase the complexity of descriptions, TACos multi-level \cite{191} expanded the TACoS \cite{188} dataset to include paragraph descriptions to instructional cooking videos. However, their dataset is constrained in the “cooking” domain and contains in the order of a 100 videos, making it unsuitable for dense-captioning of events as the models easily overfit to the training data.

Our dataset, ActivityNet Captions, aims to bridge these three orthogonal approaches by temporally annotating long videos while also building upon the complexity of descriptions. ActivityNet Captions contains videos that an average of 180s long with the longest video running to over 10 minutes. It contains a total of 100k sentences, where each sentence is temporally localized. Unlike TACoS multi-level, we have two orders of magnitude more videos and provide annotations for an open domain. Finally, we are also the first dataset to enable the study of concurrent events, by allowing our events to overlap.

### 7.7.2 Detailed dataset statistics

As noted in the main paper, the number of sentences accompanying each video is normally distributed, as seen in Figure \ref{fig:7.6a}. On average, each video contains $3.65 \pm 1.79$ sentences. Similarly, the number of words in each sentence is normally distributed, as seen in Figure \ref{fig:7.6b}. On average, each sentence contains $13.48 \pm 6.33$ words, and each video contains $40 \pm 26$ words.

There exists interaction between the video content and the corresponding temporal annotations.
Table 7.4: Compared to other video datasets, ActivityNet Captions (ANC) contains long videos with a large number of sentences that are all temporally localized and is the only dataset that contains overlapping events. (Loc. shows which datasets contain temporally localized language descriptions. Bold fonts are used to highlight the nearest comparison of our model with existing models.)

In Figure 7.7, the number of sentences accompanying a video is shown to be positively correlated with the video’s length: each additional minute adds approximately 1 additional sentence description. Furthermore, as seen in Figure 7.8, the sentence descriptions focus on the middle parts of the video more than the beginning or end.

When studying the distribution of words in Figures 7.9a and 7.9b, we found that ActivityNet Captions generally focuses on people and the actions these people take. However, we wanted to know whether ActivityNet Captions captured the general semantics of the video. To do so, we compare our sentence descriptions against the shorter labels of ActivityNet, since ActivityNet Captions annotates ActivityNet videos. Figure 7.12 illustrates that the majority of videos in ActivityNet Captions often contain ActivityNet’s labels in at least one of their sentence descriptions. We find that the many entry-level categories such as brushing hair or playing violin are extremely well represented by our captions. However, as the categories become more nuanced, such as powerbocking or cumbia, they are not as commonly found in our descriptions.

7.7.3 Dataset collection process

We used Amazon Mechanical Turk to annotate all our videos. Each annotation task was divided into two steps: (1) Writing a paragraph describing all major events happening in the videos in a paragraph, with each sentence of the paragraph describing one event (Figure 7.10a) and (2) Labeling the start
and end time in the video in which each sentence in the paragraph event occurred (Figure 7.10b.
We find complementary evidence that workers are more consistent with their video segments and
paragraph descriptions if they are asked to annotate visual media (in this case, videos) using natural
language first [118]. Therefore, instead of asking workers to segment the video first and then write
individual sentences, we asked them to write paragraph descriptions first.

Workers are instructed to ensure that their paragraphs are at least 3 sentences long where each
sentence describes events in the video but also makes a grammatically and semantically coherent
paragraph. They were allowed to use co-referencing words (ex, he, she, etc.) to refer to subjects
introduced in previous sentences. We also asked workers to write sentences that were at least 5 words
long. We found that our workers were diligent and wrote an average of 13.48 number of words per
sentence. Each of the task and examples (Figure 7.10c) of good and bad annotations.

Workers were presented with examples of good and bad annotations with explanations for what
constituted a good paragraph, ensuring that workers saw concrete evidence of what kind of work
was expected of them (Figure 7.10c). We paid workers $3 for every 5 videos that were annotated.
This amounted to an average pay rate of $8 per hour, which is in tune with fair crowd worker wage
rate [205].

Figure 7.7: Distribution of number of sentences with respect to video length. In general the longer
the video the more sentences there are, so far on average each additional minute adds one more
sentence to the paragraph.
CHAPTER 7. DENSE-CAPTIONING EVENTS IN VIDEOS

Figure 7.8: Distribution of annotations in time in ActivityNet Captions videos, most of the annotated time intervals are closer to the middle of the videos than to the start and end.

Figure 7.9: (a) The most frequently used words in ActivityNet Captions with stop words removed. (b) The most frequently used bigrams in ActivityNet Captions.

7.7.4 Annotation details

Following research from previous work that show that crowd workers are able to perform at the same quality of work when allowed to video media at a faster rate [119], we show all videos to workers at 2X the speed, i.e. the videos are shown at twice the frame rate. Workers do, however, have the option to watching the videos at the original video speed and even speed it up to 3X or 4X the speed. We found, however, that the average viewing rate chosen by workers was 1.91X while the median rate was 1X, indicating that a majority of workers preferred watching the video at its original speed. We also find that workers tend to take an average of 2.88 and a median of 1.46 times the length of the video in seconds to annotate.

At any given time, workers have the ability to edit their paragraph, go back to previous videos to make changes to their annotations. They are only allowed to proceed to the next video if this current video has been completely annotated with a paragraph with all its sentences timestamped. Changes made to the paragraphs and timestamps are saved when ”previous video or ” next video” are pressed, and reflected on the page. Only when all videos are annotated can the worker submit the task. In total, we had 112 workers who annotated all our videos.
CHAPTER 7. DENSE-CAPTIONS EVENTS IN VIDEOS

Figure 7.10: (a) Interface when a worker is writing a paragraph. Workers are asked to write a paragraph in the text box and press "Done Writing Paragraph" before they can proceed with grounding each of the sentences. (b) Interface when labeling sentences with start and end timestamps. Workers select each sentence, adjust the range slider indicating which segment of the video that particular sentence is referring to. They then click save and proceed to the next sentence. (c) We show examples of good and bad annotations to workers. Each task contains one good and one bad example video with annotations. We also explain why the examples are considered to be good or bad.
### CHAPTER 7. DENSE-CAPTIONING EVENTS IN VIDEOS

#### (a) Adding context can generate consistent captions.

<table>
<thead>
<tr>
<th>Ground Truth</th>
<th>No Context</th>
<th>With Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>A man sits with his dog in the grass and holds out discs in his hands.</td>
<td>A man is seen speaking to the camera while holding a dog around him.</td>
<td>A man is seen speaking to the camera while standing in a field with a dog.</td>
</tr>
<tr>
<td>The man balances his dog on his feet then throws Frisbee discs for him.</td>
<td>The woman continues to swing around with the frisbee as well as performing tricks.</td>
<td>The dog is seen in several clips performing tricks with his dog and running all around the yard.</td>
</tr>
<tr>
<td>The man spins his dog and holds it in his arms.</td>
<td>The man then begins to do tricks with the dog while the camera follows him.</td>
<td>The man then begins walking around with a frisbee.</td>
</tr>
<tr>
<td>Different trainers throw Frisbee discs for the dogs to catch while performing tricks.</td>
<td>A woman is seen walking out onto a field with a dog.</td>
<td>The dog runs around in circles on the field with the dog.</td>
</tr>
<tr>
<td>A woman throws discs to her dog that jumps from her back.</td>
<td>The dog jumps off the girl and the dog jumps to the dog.</td>
<td>The dog runs around in circles on the grass as he chases the frisbee.</td>
</tr>
<tr>
<td>The woman throws multiple discs in a row for her dog to catch.</td>
<td>The man spins his dog and holds it in his arms.</td>
<td>The man then begins doing tricks with the dog while the camera follows him.</td>
</tr>
</tbody>
</table>

#### (b) Comparing online versus full model

<table>
<thead>
<tr>
<th>Ground Truth</th>
<th>No Context</th>
<th>Full Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>A man is standing outside holding a black tile.</td>
<td>a man is seen speaking to the camera while holding up a tool and begins to cut.</td>
<td>a man is seen speaking to the camera while holding a tool and begins to cut.</td>
</tr>
<tr>
<td>He starts putting the tile down on the ground.</td>
<td>the man then puts a on the floor and begins putting into the tire.</td>
<td>the man then puts a on the floor and begins putting into the tire.</td>
</tr>
<tr>
<td>He cuts the tile with a red saw.</td>
<td>the man then puts a on the floor and begins putting tiles on the sides and.</td>
<td>a person is seen pushing a puck down a floor with a rag and showing the camera.</td>
</tr>
<tr>
<td>He sets chairs and flowers on the tile.</td>
<td>a person is seen pushing a puck down a floor with a rag and showing the camera.</td>
<td>the person then walks around the table and begins painting the fence.</td>
</tr>
</tbody>
</table>

#### (c) Context might add more noise to rare events.

<table>
<thead>
<tr>
<th>Ground Truth</th>
<th>No Context</th>
<th>Full Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>A little girl performs gymnastics jumping and flipping in the air.</td>
<td>A girl in a black shirt is standing on a mat.</td>
<td>The girl then begins flipping around the beam and ends by jumping off the side and walking away.</td>
</tr>
<tr>
<td>The little girl performs three back flips in the air, after she jumps.</td>
<td>A girl in a black shirt is standing on a mat.</td>
<td>The girl then flips herself over her feet and does several back flips on the mat.</td>
</tr>
<tr>
<td>The girl flips but she falls, then she stands and does cartwheels and continues doing flips and dancing.</td>
<td>A girl in a red shirt is standing in a large room in a large gymnasium.</td>
<td>The girl then flips herself over her feet and does several back flips and tricks.</td>
</tr>
</tbody>
</table>

Figure 7.11: More qualitative dense-captioning captions generated using our model. We show captions with the highest overlap with ground truth captions.
Figure 7.12: The number of videos (red) corresponding to each ActivityNet class label, as well as the number of videos (blue) that has the label appearing in their ActivityNet Captions paragraph descriptions.
Chapter 8

Conclusion

In this thesis, we have examined the research problem of leveraging large-scale visual data to study problems in both computer vision and human-computer interaction. We began by first introducing the Visual Genome dataset, which connects natural language to structured image concepts. Visual Genome provides richer annotations in the forms of labeled objects, attributes, relationships, question-answer pairs, and region descriptions than previously existing datasets. Ultimately, it is a comprehensive dataset that is well-suited to tackle a variety of new reasoning problems in computer vision.

Next, we focused on the crowdsourcing components of constructing large-scale datasets like Visual Genome. We highlighted the various crowdsourcing techniques employed in Visual Genome and discussed how these techniques are transferable to the creation of new datasets. Furthermore, we showcased a new method that produces large speedups and cost reductions in binary and categorical labeling. We also found how crowd workers remain consistent when completing microtasks for data collection, enabling us to determine good workers early in the process.

Finally, we demonstrated how the construction of new datasets enables us to develop techniques to solve more complex reasoning problems. Using Visual Genome, we demonstrated early work in a variety of new computer vision tasks, ranging from relationship prediction to generating region descriptions. Furthermore, we built a model using a newly collected video dataset that is able to automatically describe and temporally ground multiple sentence descriptions of any video. With the public release of datasets like Visual Genome and ActivityNet Captions, we expect new models to arise, allowing for greater progress in a computer’s capacity to reason about the visual world.
Bibliography


[87] Paul Hitlin. Research in the crowdsourcing age, a case study, July 2016.


[194] Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. A database for fine
grained activity detection of cooking activities. In Computer Vision and Pattern Recognition

In Mark W. Jones Xianghua Xie and Gary K. L. Tam, editors, Proceedings of the British

[196] Joel Ross, Lilly Irani, M Silberman, Andrew Zaldivar, and Bill Tomlinson. Who are the
crowdworkers?: shifting demographics in mechanical turk. In CHI’10 extended abstracts on

[197] Sascha Rothe and Hinrich Schütze. Autoextend: Extending word embeddings to embeddings

[198] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng
Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei.
ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision
(IJCV), pages 1–42, April 2015.

[199] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng
Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C Berg, and Fei-Fei Li.
Imagenet large scale visual recognition challenge. International Journal of Computer Vision,
pages 1–42, 2014.

[200] Olga Russakovsky, Li-Jia Li, and Li Fei-Fei. Best of both worlds: human-machine collaboration
for object annotation. In Proceedings of the IEEE Conference on Computer Vision and Pattern

[201] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a
database and web-based tool for image annotation. International journal of computer vision,

and output. In Proceedings of the 25th annual ACM symposium on User interface software

and question answering by visual verification of relation phrases. In Proceedings of the IEEE

Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pages 1745–1752. IEEE,
2011.


