Supplementary material for
Best of both worlds: human-machine collaboration for object annotation

Olga Russakovsky[1] Li-Jia Li[2] Li Fei-Fei[1]
[1]Stanford University [2]Snapchat*

1. Introduction

We provide additional details about our principled human-in-the-loop framework for efficiently detecting all objects in an image.

Section 2 provides details about how user feedback is used in calculating probabilities (referenced in **Section 4.3 in main paper**). We first show how each question posed to users affects multiple probability estimates about the image. We then discuss probability updates in some special cases: in case of open-ended questions (such as asking the user to draw a bounding box) and in case of computing the probabilities of there being more class instances or new objects in the image.

Section 3 discusses the procedure for probability calibration of the outputs of computer vision image classifiers and object detectors (referenced in **footnote 5 of main paper**).

Section 4 talks about the design of user interfaces for collecting human feedback (referenced in **caption of Figure 3 of main paper**). We show the interfaces themselves in Figures 1-9.

Section 5 describes how we semi-automatically generate sets of positive and negative examples for computing the human error rates (referenced in **footnote 7 of main paper**).

2. Using user feedback in the human-in-the-loop probabilistic framework

As discussed in Section 4.3 of the main paper, our set of user inputs $\mathcal{U}$ contains multiple types of information. Our goal is to estimate $P(u_t|E_k^T)$ where u_t is a user response to some question a_t and E_k^T is a fact about the image related to some other question a_T (in particular Eqn. (6) of the main paper).

Types of events E_k^T. Based on the tasks described in Table 1 of the main paper, we consider 5 types of E_k^T:

1. det(B, C) for whether box B is correct around an instance of class C (**verify-box** task)
2. cls(C) for whether class C is present in the image (**verify-image** task)
3. moreinst$(\mathcal{B}, C)$ for whether there are more instances of class C besides those in boxes $\mathcal{B}$ (**verify-cover** and **draw-box** tasks)
4. morecls$(\mathcal{C})$ for whether there are more object classes in the image $\mathcal{C}$ (for **name-image** task)
5. obj(B) for whether box B is a tight box around *some* object (for **verify-object** and **name-box** tasks)

*This work was done while Li-Jia Li was at Yahoo! Labs

Computing the *true* answer to a question. In order to make a decision about whether the user made a mistake or not in u_t, we first need to determine the true answer to the question a_t given that event E_k^T happened. This is shown in Table 1. An event E_k^T sometimes determines the correct answers to more than one question.

For example, consider the event E_k^T of class C not present in the image (this is event "cls(C) = 0" in Table 1). The true answer is then determined to be "no" to three questions: (1) when asked if C is contained in the image (verify-image), (2) when asked if C is contained in any box B (verify-box), and (3) when asked if there are more instances of class C in the image (verify-cover). For other questions we have no information about the true answer given the event E_k^T.

Judging user input as correct, wrong or undecided Having computed the true answers, we can now judge the user input u_t in response to a question a_t. An answer is judged as *wrong* if it doesn't match the correct answer in Table 1. An answer is judged as *correct* if it matches the correct answer in Table 1 *and* the event corresponds precisely to the question, as shown by the shaded boxes in Table 1. All other answers are judged as *undecided*.

To understand the extra layer of complication with judging correct answers, consider the influence of the question "does box B contain an instance of class C" (verify-box) on the probability of the object class C being in the image. If

978-1-4673-6964-0/15/$31.00 ©2015 IEEE

Event E_k^T	True answer to each question type a_t given E_k^T				
	Verify-box: is box B for class C?	**Verify-image:** is class C in image?	**Verify-cover:** are there more instances of class C besides in $\mathcal{B}'$? (same for **draw-box**)	**Name-image:** name another object in image besides $\mathcal{C}'$?	**Verify-object:** is there an object in box B? (similarly for **name-object**)
$\text{det}(B, C) = 1$	✓	✓	✓ if $B \notin \mathcal{B}'$	–	✓
$\text{det}(B, C) = 0$	✗	–	–	–	–
$\text{cls}(C) = 1$	–	✓	–	–	–
$\text{cls}(C) = 0$	✗	✗	✗	–	–
$\text{moreinst}(\mathcal{B}, C) = 1$	–	✓	✓ if $\mathcal{B}' \subseteq \mathcal{B}$	–	–
$\text{moreinst}(\mathcal{B}, C) = 0$	✗ if $B \notin \mathcal{B}$	–	✗ if $\mathcal{B} \subseteq \mathcal{B}'$	–	–
$\text{morecls}(\mathcal{C}) = 1$	–	–	–	✓ if $\mathcal{C}' \subseteq \mathcal{C}$	–
$\text{morecls}(\mathcal{C}) = 0$	–	✗ if $C \notin \mathcal{C}$	–	✗ if $\mathcal{C} \subseteq \mathcal{C}'$	–
$\text{obj}(B) = 1$	–	–	–	–	✓
$\text{obj}(B) = 0$	✗	–	–	–	✗

Table 1. For every event E (row) and question (column), the table reports what the *true* answer to the question if the event E happened. ✓ means "yes" is the true answer, ✗ means "no" is the true answer, and – means that event E provides no information about the true answer. – is the default when not specified. Here every question is treated as a binary question: for example, for **draw-box** question, the answer of drawing a box is simply "yes" and the answer of refusing to draw a box is "no".

the answer to the question is "yes," then this can only happen if class C is in the image and is certain to be the wrong answer otherwise. There is a direct influence. If the answer is "no," then it might be correct whether or not class C is in the image. To simplify the computation, we then judge the answer as undecided.

User input accuracy probabilities. Finally, after the answers are all judged as correct, wrong or undecided, we incorporate them back into the probability computation.

Every user input u_t is obtained in response to a question. Each input u_t is then associated with a probability β_t which depends on the average user accuracy rate for this type of question. Let E_0^t correspond to the answer "no" and E_1^t correspond to "yes". Similarly, let $u_t = 1$ if user says "yes" and $u_t = 0$ if user says "no". Then $\beta_t = P(u_t = j|E_j^t)$ for $j \in \{0, 1\}$. The accuracy probabilities β_t are shown in Table 2 of the main paper.

We can then compute $P(u_t|E_j^t) = \beta_t$ if answer u_t is *correct*, $P(u_t|E_j^t) = 1 - \beta_t$ if answer u_t is *wrong* , and use the prior from Eqn. (5) of the main paper otherwise..

2.1. Special cases

Here we briefly note some exceptions to the computation described above.

More instances computation. Recall that computing complicated events such as $P(\text{moreinst}(\mathcal{B}, C)|I)$ (Eqn. (8) of main paper) relies on additionally on detection probabilities $P(\text{det}(B, C)|I)$ as well as image classification probabilities $P(\text{cls}(C)|I)$ (Section 4.3.1 of the main paper). To effectively utilize all user input $\mathcal{U}$ in this computation, we break up the set $\mathcal{U}$ into $\mathcal{U}_1$, which is the set of user input directly relevant to $\text{moreinst}(\mathcal{B}, C)$ and $\mathcal{U}_2$, which is all other input. Then have

$$P(\text{moreinst}(\mathcal{B}, C)|I, \mathcal{U}) \propto P(\text{moreinst}(\mathcal{B}, C)|I, \mathcal{U}_2)P(\mathcal{U}_1|\text{moreinst}(\mathcal{B}, C)) \quad (1)$$

$\mathcal{U}_1$ consists of all user input obtained from verify-cover or draw-box tasks that we can judge based on the event $\text{moreinst}(\mathcal{B}, C)$ (Table 1). $\mathcal{U}_2$ is all other user input. We use $P(\text{det}(B, C) = 1|I, \mathcal{U}_2)$ and $P(\text{cls}(B, C) = 1|I, \mathcal{U}_2)$ in this computation, which allows us to successfully incorporate the entire set of user input $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$.

The computation for $P(\text{morecls}(\mathcal{C}))$ is similar.

New object computation. The computation for $P(\text{newobj}(B)|I,\mathcal{U})$ also requires breaking up $\mathcal{U}$ into two parts, but in a slightly different way. Generalizing Eqn. (9) of the main paper we have

$$\begin{aligned} P(\text{newobj}(B)|I,\mathcal{U}) =& P(\text{obj}(B)|I,\mathcal{U}_1)\times \\ & \prod_C (1 - P(\text{det}(B,C)|I,\mathcal{U}_2)) \end{aligned} \tag{2}$$

with $\mathcal{U}_1$ is all input from verify-object or name-object tasks related to box obj(B) (Table 1) and $\mathcal{U}_2 = \mathcal{U} - \mathcal{U}_1$. This is to ensure that the independence assumption is not violated.

Open-ended questions For open-ended question such as draw-box we have to consider both the probability that the user draws a box when there is one and that the drawn box is indeed a good box around an object instance. The latter is noted in Equation (7) of the main paper as $P(\hat{E}|u_{\hat{T}})$, where $u_{\hat{T}}$ is the fact that user drew the box at time $\hat{T}$ and $\hat{E}$ is fact that the box is correct. We stated that this is computed from user error rates which is true but somewhat subtle: computing this error rate directly would be influenced by the distribution of positive and negative images shown to the user (positive images being the ones that indeed contain an unannotated instance). To avoid this problem we add an extra variable pos corresponding to the fact that the image is positive, so an unannotated box indeed existed in the image. Then $P(\hat{E}|u_{\hat{T}}) = P(\hat{E}, pos|u_{\hat{T}})$. So

$$P(\hat{E}|u_{\hat{T}}) \propto P(\hat{E}|u_{\hat{T}}, pos)P(u_{\hat{T}}|pos)P(pos) \tag{3}$$

Now $P(\hat{E}|u_{\hat{T}}, pos)$ can be estimated as the probability that a bounding box that the user drew on a positive image is indeed correct (reported in caption of Table 2 of the main paper), $P(u_{\hat{T}}|pos)$ is the true positive accuracy for the draw-box task (reported in Table 2 of the main paper), and $P(pos)$ is the current estimate of the image being positive.

3. Details about probability calibration

In order to use computer vision models in our framework, we need to obtain accurate probability estimates from the models. The output of object detectors and image classifiers x is commonly converted to a probability by fitting a 2-parameter sigmoid

$$p = \frac{1}{1 + \exp(a_1 x + a_2)} \tag{4}$$

to the CNN output on the ILSVRC val1 set [1]. We bin the detections from all classes into 20 bins in increments of 5% confidence, and compute the error between the expected probability produced by the model (2.5%, 7.5%, 12.5%, etc.) and the actual fraction of positive examples in that bin. The average absolute probability error is 8.7% on ILSVRC val2 set due to the model being overconfident on the high-scoring examples. To compensate, we learn the 3rd parameter $a_3 \in [0, 1]$ with

$$p = \frac{a_3}{1 + \exp(a_1 x + a_2)} \tag{5}$$

to allow the model to automatically estimate its level of confidence. This reduced the error down to 5.2%. More accurate models such as [2] can also be used.

4. Human annotation design and observations

UI overview. Figure 1 shows a zoomed-out view of the annotation interface. The general instructions at the bottom of the page (shown in Figure 2) discuss what is considered a good bounding box (with examples) and what types of objects should be annotated. These instructions are shown to workers at beginning of the work and are always available for reference at the bottom of the page. Each type of task has a separate interface with brief specific instructions, shown in Figures 3-9.

Quality control. The questions are presented to users in batches of 20-25 questions. Each batch contains 4-5 “gold standard” questions. These are questions which were verified by trusted subjects (previously unfamiliar with our UI) who deemed that the correct answer should be “obvious” to a careful annotator. When deployed, the annotation UI prevents users from submitting their work if they incorrectly answer more than 1 gold standard question.

The UI also had additional sanity checks built in. First, it prevents users from drawing a bounding box around an object instance if it is too close to a known bounding box. Second, it requires that users spend at least 1 second on each question. In our in-house experiments with trusted workers, this was the minimum amount of time necessary to answer a question correctly. This control was implemented since the interface has keyboard shortcuts (1 and 2 to answer “yes” or “no”, and left and right arrows to move between questions) which makes it possible for spammers to potentially blindly answer all 20 questions in just a few seconds (if all questions are binary).

Cost. On Amazon Mechanical Turk we pay workers 10 cents to answer 20 questions. Since on average it takes 7.69 seconds to answer a question (Table 2 in main paper) this comes out to $2.34 per hour.

To simulate the real use case (where different types of questions are automatically generated by our system out of order) we assigned a *random* selection of tasks to each batch. This slowed down the worker responses since it required reading multiple sets of instructions. Some workers even complained to us via email about this.[1] One interesting

Task	Cost (seconds)	
	Positive response	Negative response
Verify-image	5.64	4.88
Verify-box	6.22	5.45
Verify-cover	7.47	7.63
Draw-box	12.34	8.67
Verify-object	6.08	5.21
Name-object	12.19	7.09
Name-image	12.53	7.67

Table 2. Cost of each task (in median human time) broken down by positive versus negative responses. On average, positive responses take longer than negative ones. One interesting potential extension to our human-machine object annotation framework would be to incorporate this fact.

extension to our current human-machine object annotation framework would be to consider the reduced human cost if asking multiple questions of the same type consecutively.

Finally, we observed that for many types of questions answering positively took longer than answering negatively (especially for the open-ended questions). Table 2 documents this. Another potential extension in our framework would be to incorporate this fact when making decisions about the optimal next question to ask the annotators.

5. Generating positive and negative examples for computing human error rates

In order to estimate human error rates, we perturb the annotations from ILSVRC detection val1 set to obtain a representative positive and negative examples.

For most tasks, this is straight-forward. For example, for verify-box task ("is B a good box around an instance of class C?") we generate the positive set by sampling ground truth boxes for this class and perturbing them to between 0.7 and 1 IOU. We generate the negative set by sampling boxes between 0 and 0.5 IOU, as well as boxes corresponding to other object classes.

However, the negative sets of verify-object ("is there an object in box B?"), name-object ("name the object in box B") and name-image ("name another object in this image") tasks can't be generated automatically: there are only 200 classes labeled in ILSVRC, and so any randomly generated region or image from ILSVRC might accidentally contain some other object class beyond the annotated 200. Thus, we sampled some likely negative candidates from ILSVRC and asked 4 AMT workers to determine if there is an additional object in there. If 3 or more workers said that the box does not contain an object, we manually verified it to confirm and then included it in the negative set.

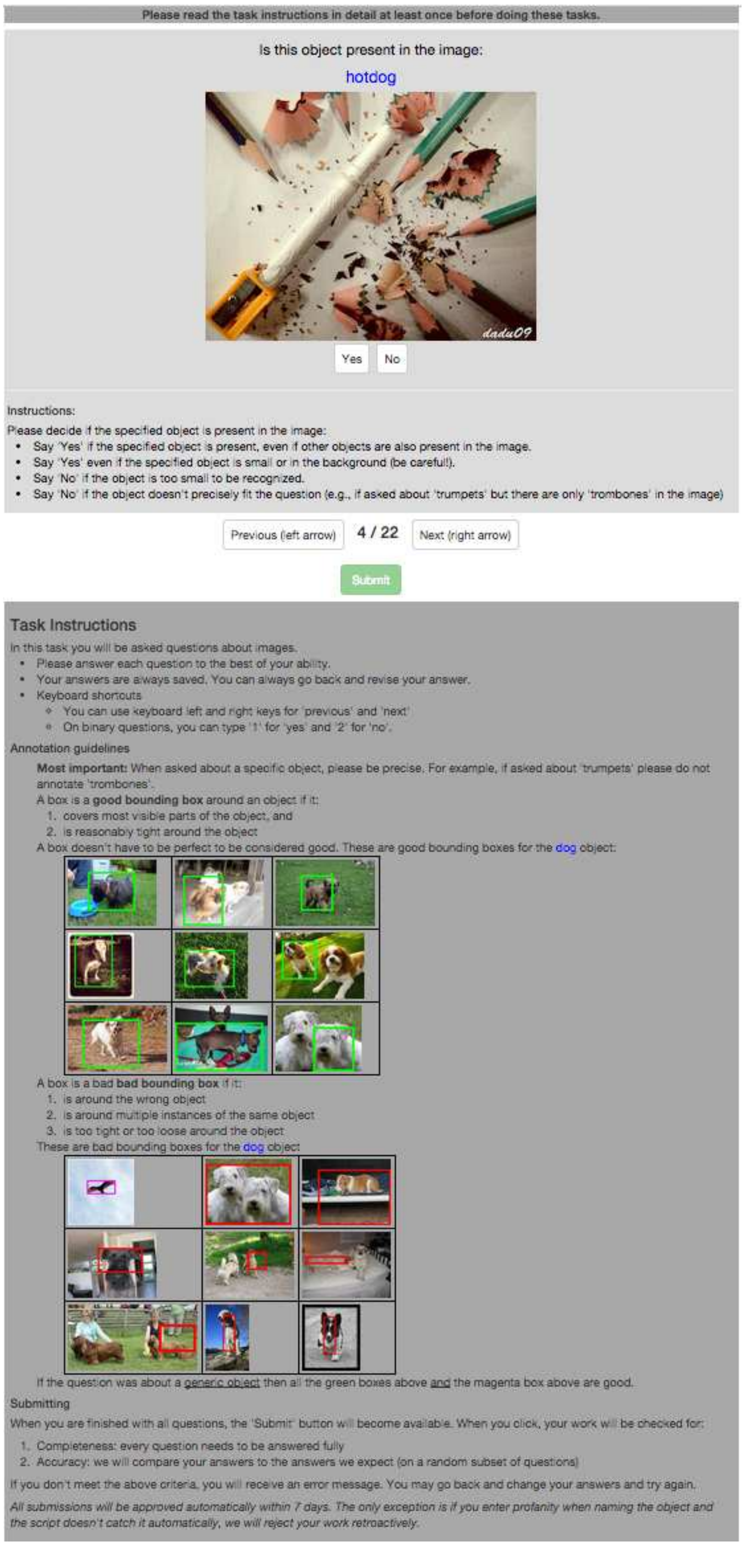
The likely negative candidates for verify-object and name-object were simply random regions on the image that had overlap less than 0.3 with any annotated box. The likely negative candidates for name-image were generated by selecting images where the annotated bounding boxes cover more than 90% of the image area.

[1]Given that the random selection of questions was necessary in our setting, we attempted to at least simplify the process for the labeler as much as possible by arranging the question types in logical order, roughly from "easiest" to "hardest": verify-image, name-image, verify-box, draw-box, verify-cover, verify-object, name-object. This is to help the annotators to slowly familiarize themselves with the questions; this was proven effective in our in-house annotation experiments.

References

[1] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*, 1999. 3

[2] W. Scheirer, N. Kumar, P. N. Belhumeur, and T. E. Boult. Multi-attribute spaces: Calibration for attribute fusion and similarity search. In *Computer Vision and Pattern Recognition (CVPR)*, 2012. 3



Please read the task instructions in detail at least once before doing these tasks.

Is this object present in the image:

hotdog

Instructions:

Please decide if the specified object is present in the image:

- Say 'Yes' if the specified object is present, even if other objects are also present in the image.
- Say 'Yes' even if the specified object is small or in the background (be careful!).
- Say 'No' if the object is too small to be recognized.
- Say 'No' if the object doesn't precisely fit the question (e.g., if asked about 'trumpets' but there are only 'trombones' in the image)

Task Instructions

In this task you will be asked questions about images.

- Please answer each question to the best of your ability.
- Your answers are always saved. You can always go back and revise your answer.
- Keyboard shortcuts
  - You can use keyboard left and right keys for 'previous' and 'next'
  - On binary questions, you can type '1' for 'yes' and '2' for 'no'.

Annotation guidelines

Most important: When asked about a specific object, please be precise. For example, if asked about 'trumpets' please do not annotate 'trombones'.

A box is a **good bounding box** around an object if it:

1. covers most visible parts of the object, and
2. is reasonably tight around the object

A box doesn't have to be perfect to be considered good. These are good bounding boxes for the dog object:

A box is a bad **bad bounding box** if it:

1. is around the wrong object
2. is around multiple instances of the same object
3. is too tight or too loose around the object

These are bad bounding boxes for the dog object

If the question was about a generic object then all the green boxes above and the magenta box above are good.

Submitting

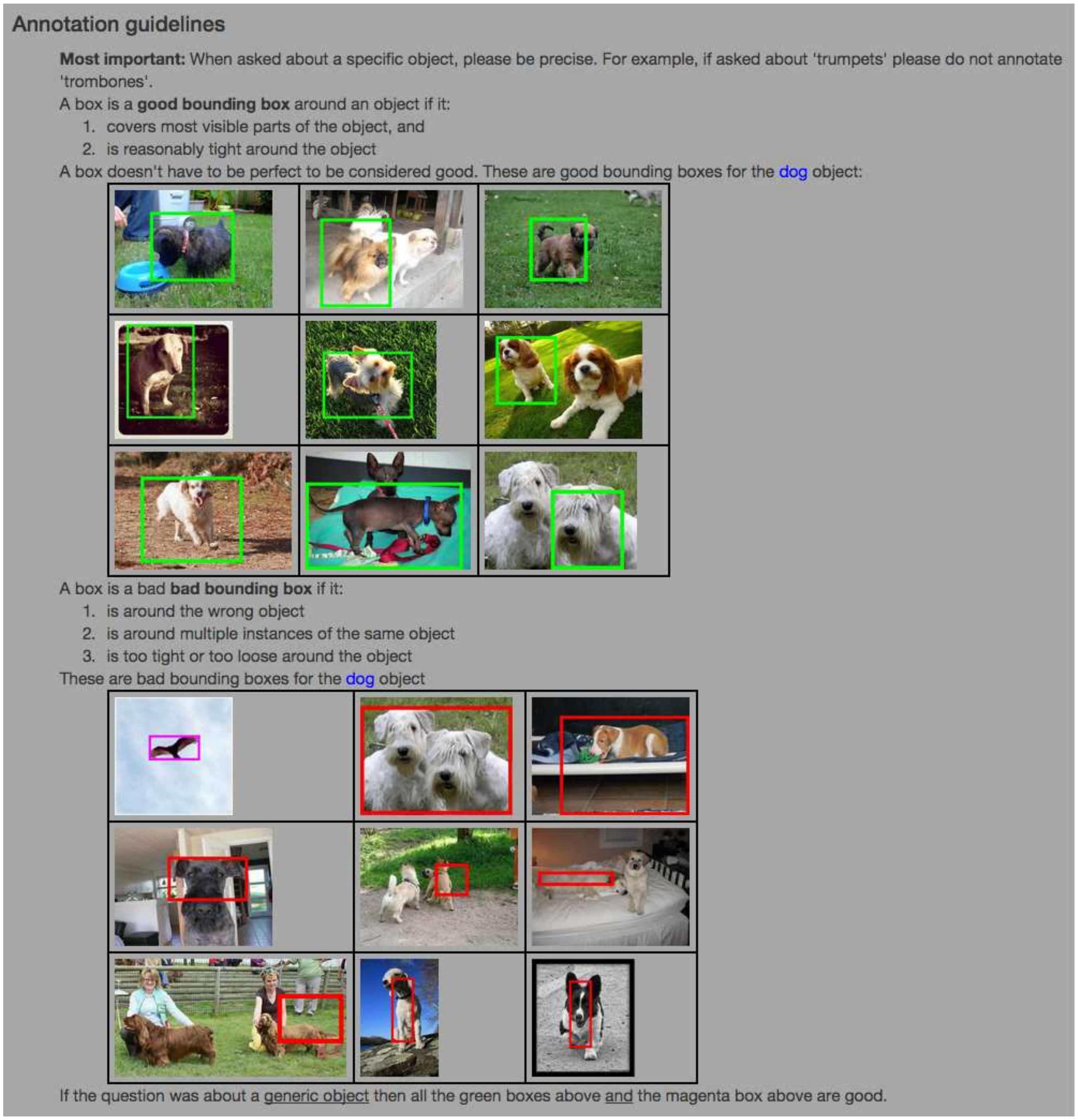
When you are finished with all questions, the 'Submit' button will become available. When you click, your work will be checked for:

1. Completeness: every question needs to be answered fully
2. Accuracy: we will compare your answers to the answers we expect (on a random subset of questions)

If you don't meet the above criteria, you will receive an error message. You may go back and change your answers and try again.

All submissions will be approved automatically within 7 days. The only exception is if you enter profanity when naming the object and the script doesn't catch it automatically, we will reject your work retroactively.

Figure 1. Overview of our annotation layout. A close-up of the instructions is in Figure 2. Only one type of task is shown here; closeups of each of the seven types are in Figures 3-9).



Annotation guidelines

Most important: When asked about a specific object, please be precise. For example, if asked about 'trumpets' please do not annotate 'trombones'.

A box is a **good bounding box** around an object if it:

1. covers most visible parts of the object, and
2. is reasonably tight around the object

A box doesn't have to be perfect to be considered good. These are good bounding boxes for the dog object:

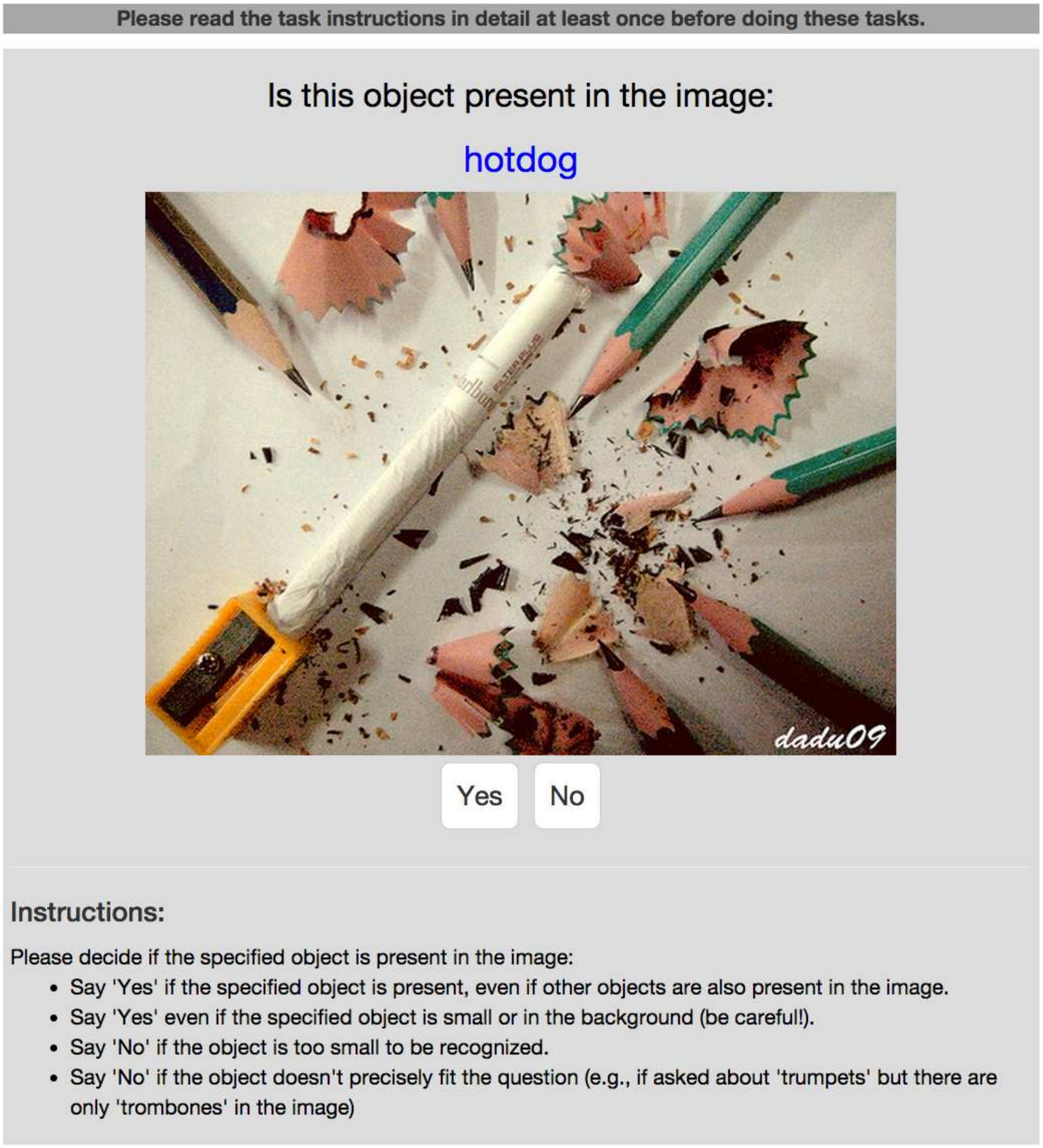
A box is a bad **bad bounding box** if it:

1. is around the wrong object
2. is around multiple instances of the same object
3. is too tight or too loose around the object

These are bad bounding boxes for the dog object

If the question was about a generic object then all the green boxes above and the magenta box above are good.

Figure 2. General instructions for our human annotation tasks.



Please read the task instructions in detail at least once before doing these tasks.

Is this object present in the image:

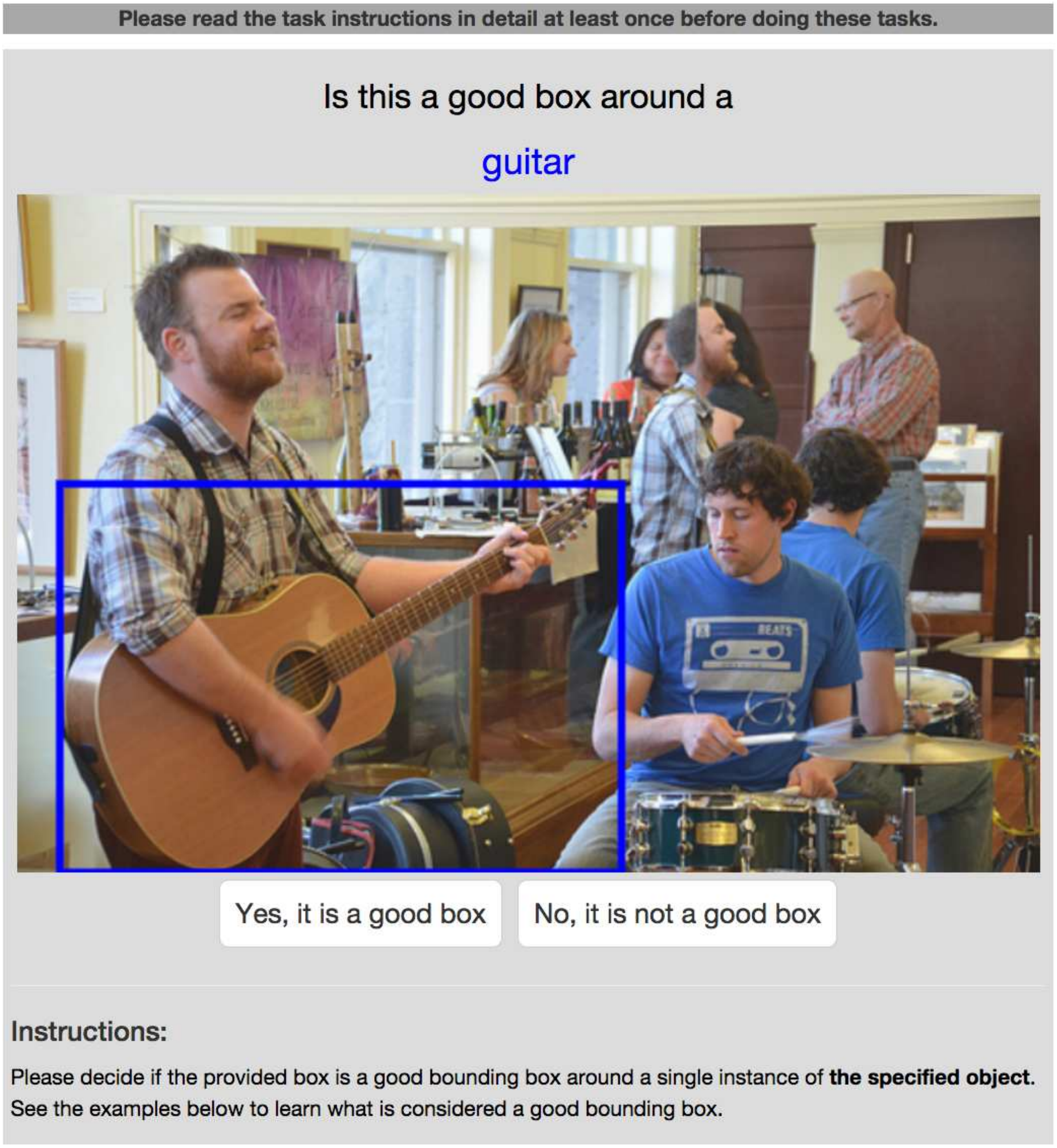
hotdog

Instructions:

Please decide if the specified object is present in the image:

- Say 'Yes' if the specified object is present, even if other objects are also present in the image.
- Say 'Yes' even if the specified object is small or in the background (be careful!).
- Say 'No' if the object is too small to be recognized.
- Say 'No' if the object doesn't precisely fit the question (e.g., if asked about 'trumpets' but there are only 'trombones' in the image)

Figure 3. User interface for **verify-image** task. The correct answer is "no."


Please read the task instructions in detail at least once before doing these tasks.

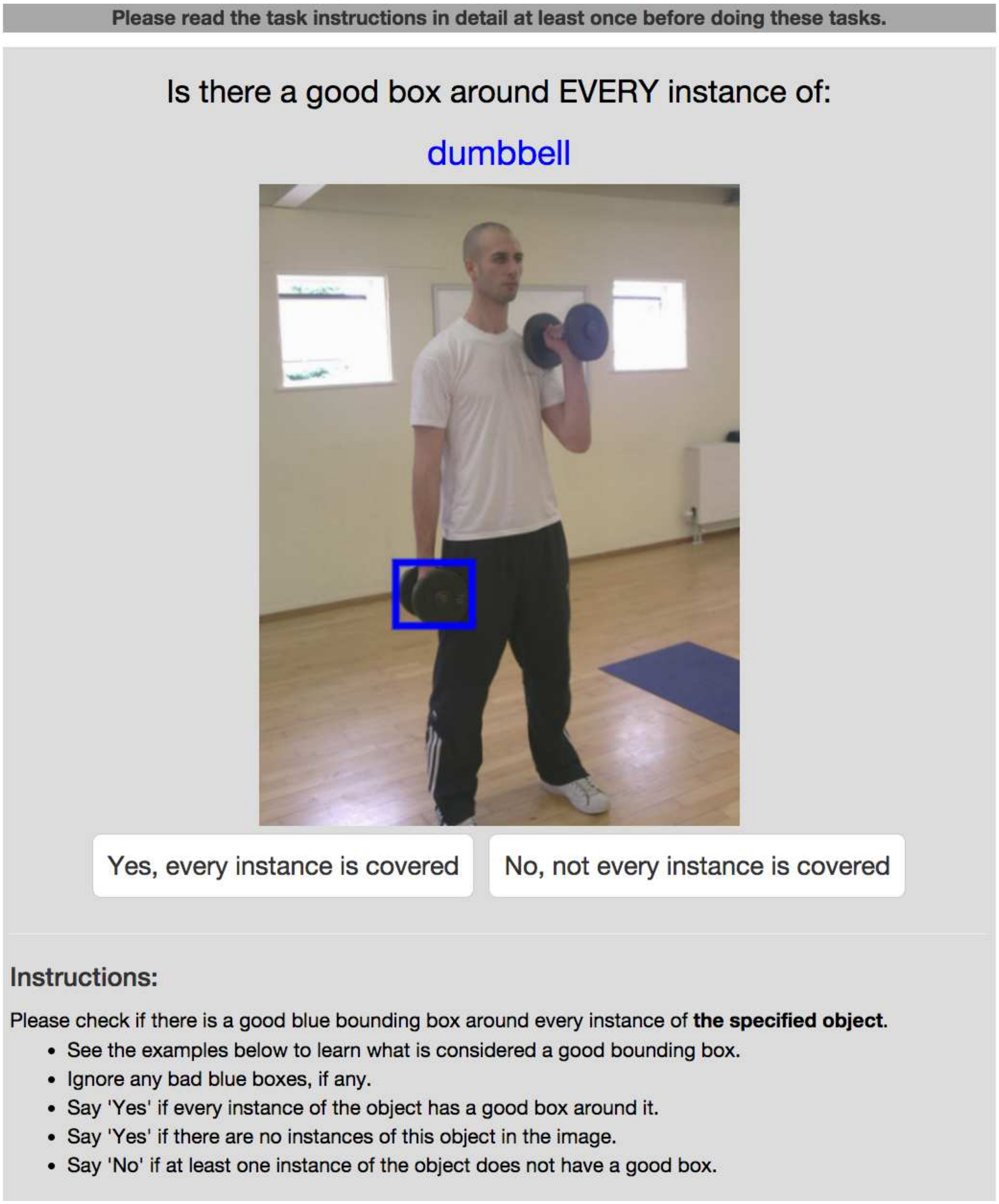
Is this a good box around a

guitar

Instructions:

Please decide if the provided box is a good bounding box around a single instance of **the specified object**. See the examples below to learn what is considered a good bounding box.

Figure 4. User interface for **verify-box** task. The correct answer is “yes.”



Please read the task instructions in detail at least once before doing these tasks.

Is there a good box around EVERY instance of:

dumbbell

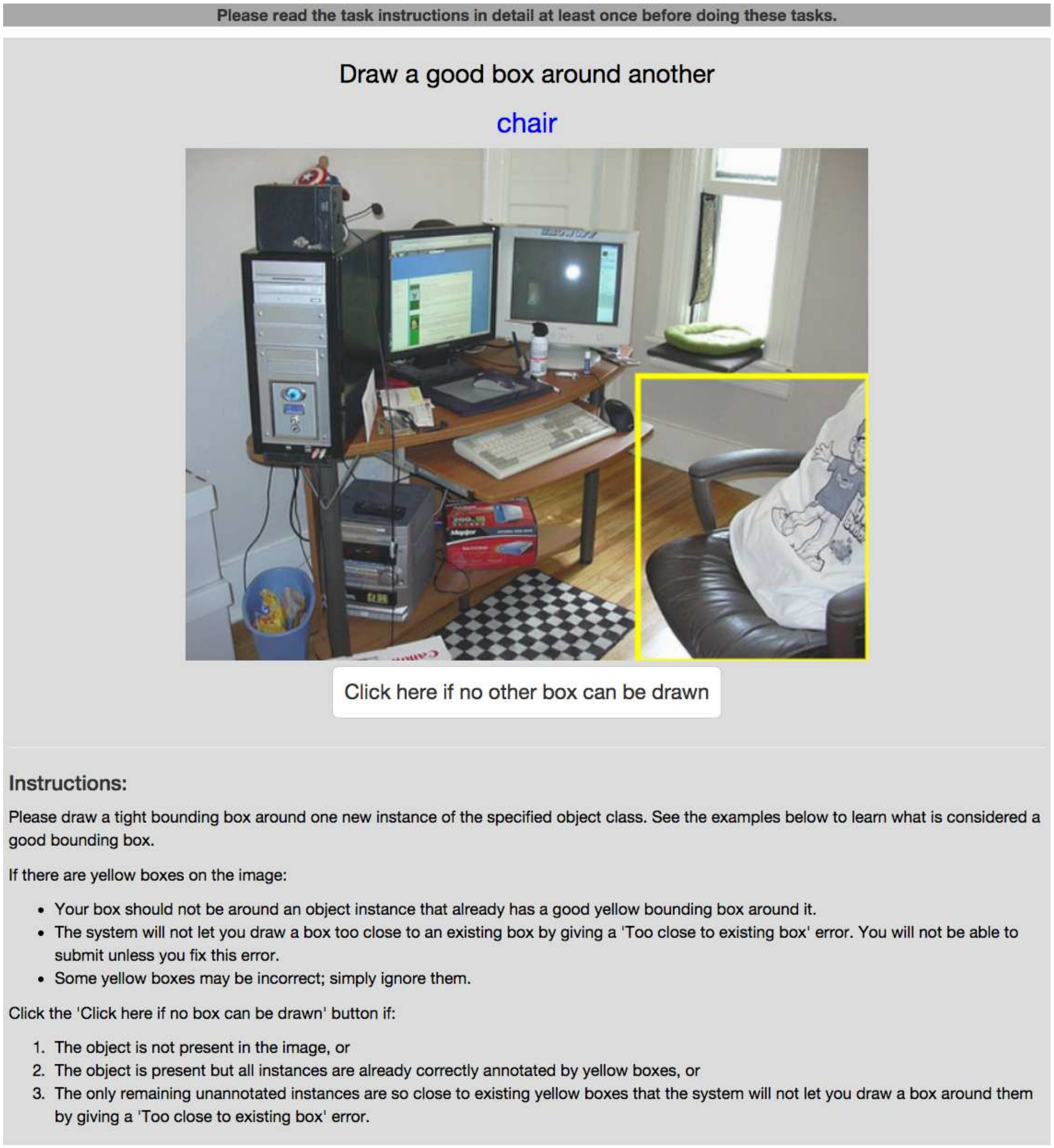
Yes, every instance is covered
No, not every instance is covered

Instructions:

Please check if there is a good blue bounding box around every instance of **the specified object**.

- See the examples below to learn what is considered a good bounding box.
- Ignore any bad blue boxes, if any.
- Say 'Yes' if every instance of the object has a good box around it.
- Say 'Yes' if there are no instances of this object in the image.
- Say 'No' if at least one instance of the object does not have a good box.

Figure 5. User interface for **verify-cover** task. The correct answer is "no."



Please read the task instructions in detail at least once before doing these tasks.

Draw a good box around another

chair

Instructions:

Please draw a tight bounding box around one new instance of the specified object class. See the examples below to learn what is considered a good bounding box.

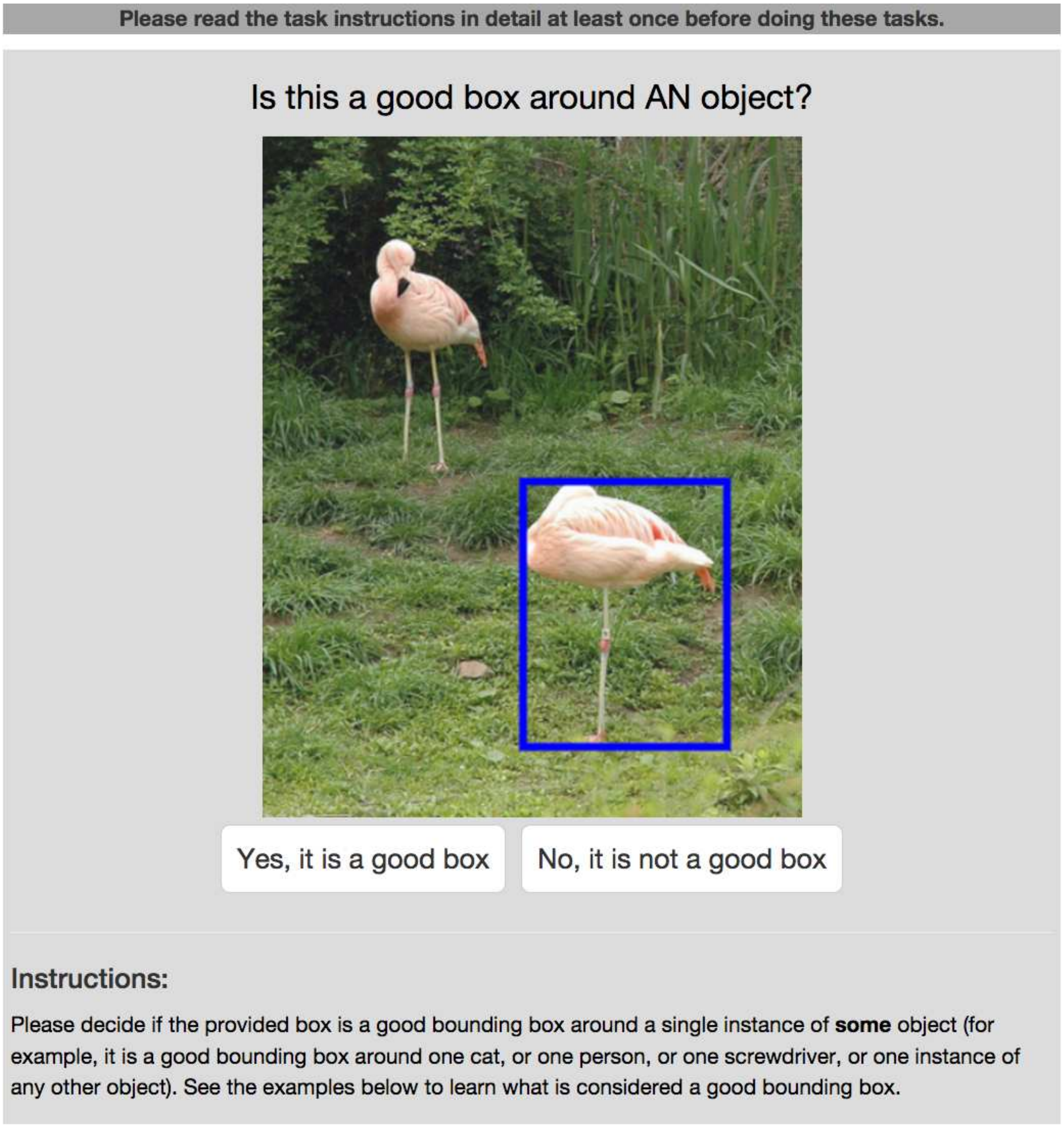
If there are yellow boxes on the image:

- Your box should not be around an object instance that already has a good yellow bounding box around it.
- The system will not let you draw a box too close to an existing box by giving a 'Too close to existing box' error. You will not be able to submit unless you fix this error.
- Some yellow boxes may be incorrect; simply ignore them.

Click the 'Click here if no box can be drawn' button if:

1. The object is not present in the image, or
2. The object is present but all instances are already correctly annotated by yellow boxes, or
3. The only remaining unannotated instances are so close to existing yellow boxes that the system will not let you draw a box around them by giving a 'Too close to existing box' error.

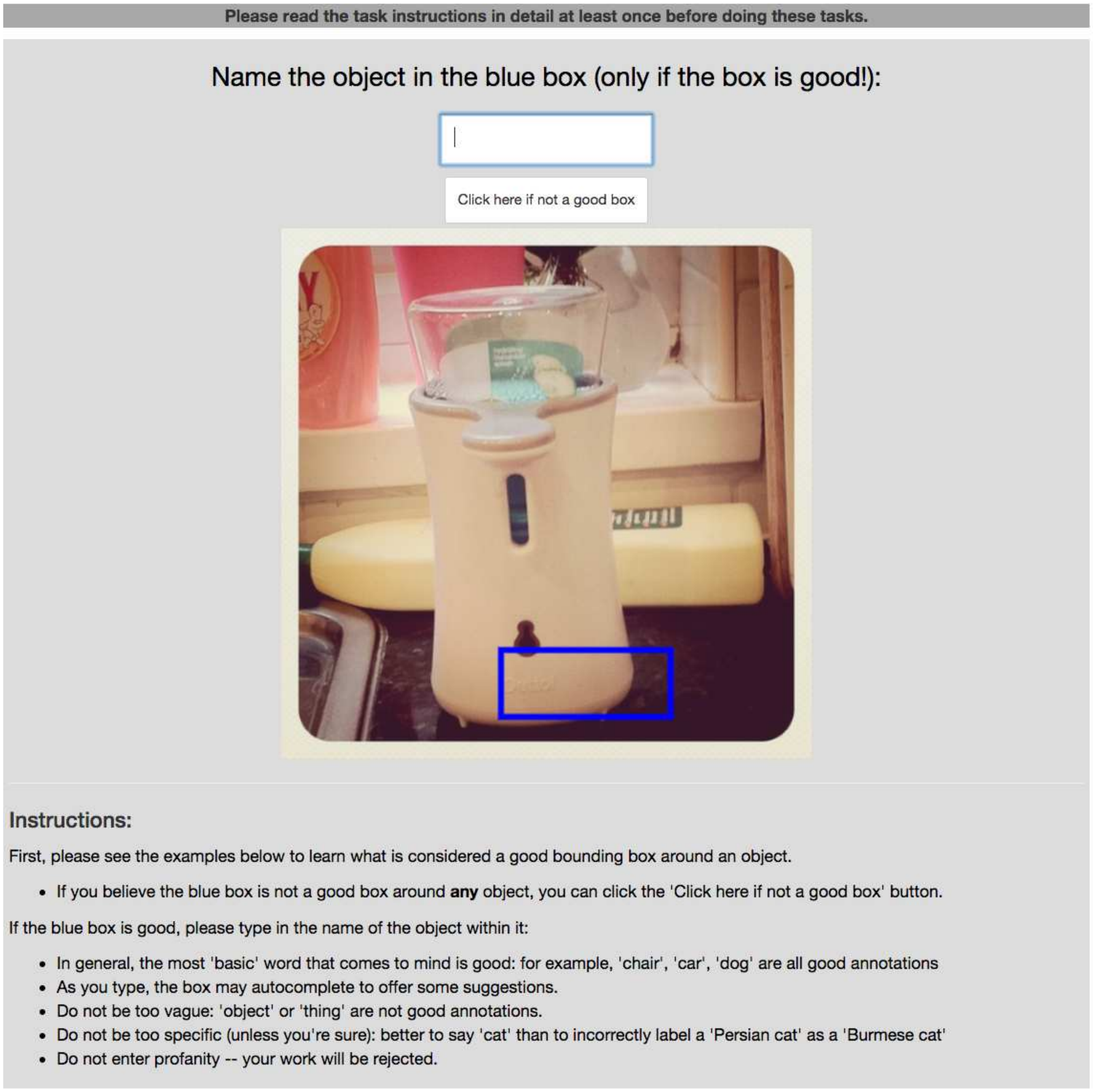
Figure 6. User interface for **draw-box** task. The correct answer is “no other box can be drawn.”


Please read the task instructions in detail at least once before doing these tasks.
Is this a good box around AN object?
Yes, it is a good box
No, it is not a good box

Instructions:

Please decide if the provided box is a good bounding box around a single instance of **some** object (for example, it is a good bounding box around one cat, or one person, or one screwdriver, or one instance of any other object). See the examples below to learn what is considered a good bounding box.

Figure 7. User interface for **verify-object** task. The correct answer is "yes."



Please read the task instructions in detail at least once before doing these tasks.

Name the object in the blue box (only if the box is good!):

Instructions:

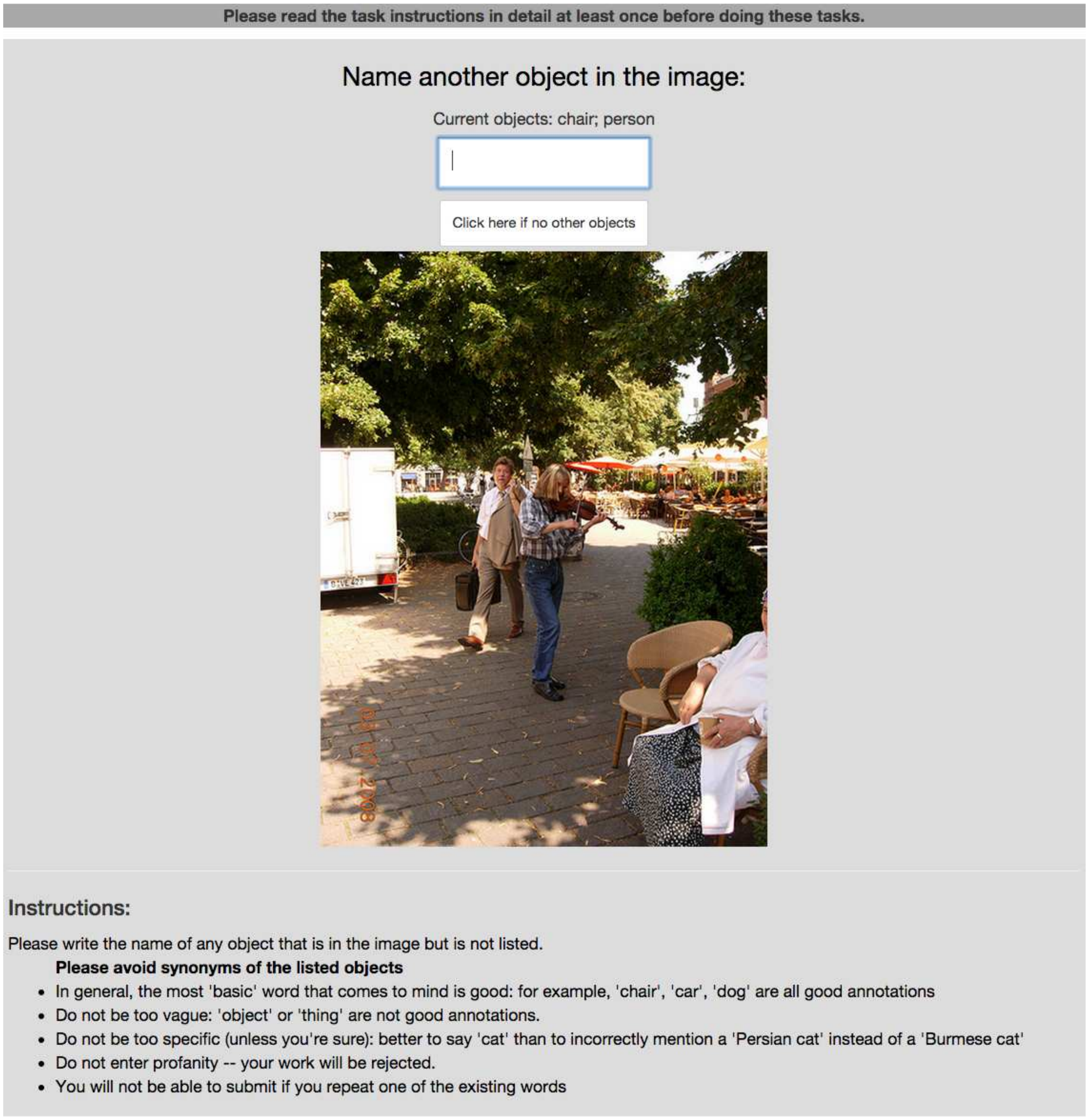
First, please see the examples below to learn what is considered a good bounding box around an object.

- If you believe the blue box is not a good box around **any** object, you can click the 'Click here if not a good box' button.

If the blue box is good, please type in the name of the object within it:

- In general, the most 'basic' word that comes to mind is good: for example, 'chair', 'car', 'dog' are all good annotations
- As you type, the box may autocomplete to offer some suggestions.
- Do not be too vague: 'object' or 'thing' are not good annotations.
- Do not be too specific (unless you're sure): better to say 'cat' than to incorrectly label a 'Persian cat' as a 'Burmese cat'
- Do not enter profanity -- your work will be rejected.

Figure 8. User interface for **name-object** task. The correct answer is "not a good box."



Please read the task instructions in detail at least once before doing these tasks.

Name another object in the image:

Current objects: chair; person

Instructions:

Please write the name of any object that is in the image but is not listed.

Please avoid synonyms of the listed objects

- In general, the most 'basic' word that comes to mind is good: for example, 'chair', 'car', 'dog' are all good annotations
- Do not be too vague: 'object' or 'thing' are not good annotations.
- Do not be too specific (unless you're sure): better to say 'cat' than to incorrectly mention a 'Persian cat' instead of a 'Burmese cat'
- Do not enter profanity -- your work will be rejected.
- You will not be able to submit if you repeat one of the existing words

Figure 9. User interface for **name-image** task. The correct answer is, for example, “umbrella.”