Efficient Object Annotation via Speaking and Pointing

Michael Gygli · Vittorio Ferrari

Abstract Deep neural networks deliver state-of-the-art visual recognition, but they rely on large datasets, which are time-consuming to annotate. These datasets are typically annotated in two stages: (1) determining the presence of object classes at the image level and (2) marking the spatial extent for all objects of these classes. In this work we use speech, together with mouse inputs, to speed up this process. We first improve stage one, by letting annotators indicate object class presence via speech. We then combine the two stages: annotators draw an object bounding box via the mouse and simultaneously provide its class label via speech. Using speech has distinct advantages over relying on mouse inputs alone. First, it is fast and allows for direct access to the class name, by simply saying it. Second, annotators can simultaneously speak and mark an object location. Finally, speech-based interfaces can be kept extremely simple, hence using them requires less mouse movement compared to existing approaches. Through extensive experiments on the COCO and ILSVRC datasets we show that our approach yields high-quality annotations at significant speed gains. Stage one takes $2.3\times - 14.9\times$ less annotation time than existing methods based on a hierarchical organization of the classes to be annotated. Moreover, when combining the two stages, we find that object class labels come for free: annotating them at the same time as bounding boxes has zero additional cost. On COCO, this makes the overall process $1.9\times$ faster than the two-stage approach.

Keywords Speech-based annotation · Object annotation · Multimodal interfaces · Large-scale computer vision

M. Gygli ✉
Google Research
E-mail: gyglim@google.com

V. Ferrari
Google Research
E-mail: vittoferrari@google.com

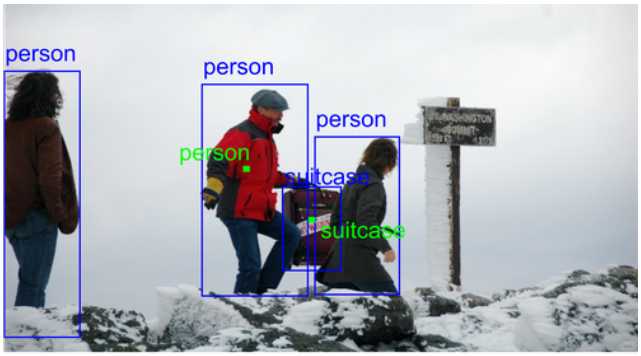


Fig. 1: Illustration of common stages of image annotation: annotators first provide object class labels at the image level (Russakovsky et al. 2015a; Kuznetsova et al. 2018) (red), sometimes associated to a specific object via a click as in (Lin et al. 2014) and our approach for object class labelling (green). Following stages then annotate the spatial extent of all objects of these classes, *e.g.* with bounding boxes or segmentations (blue). Speech provides a natural way to combine the two stages to simultaneously annotate class labels and bounding boxes.

1 Introduction

Deep neural networks need millions of training examples to obtain high performance. Large and diverse datasets such as ILSVRC (Russakovsky et al. 2015a), COCO (Lin et al. 2014) or Open Images (Kuznetsova et al. 2018) therefore lie at the heart of the breakthrough and ongoing advances in visual recognition.

Datasets for recognition are typically annotated in two stages (Russakovsky et al. 2015a; Kuznetsova et al. 2018; Lin et al. 2014; Su et al. 2012) (Fig. 1): (i) determining the presence or absence of object classes in each image, and (ii) providing bounding boxes or segmentation masks for all objects of the classes present.

Designing interfaces for stage one, which we call *object class labelling*, has traditionally been challenging and their use time-consuming. The key question is how to quickly navigate a vocabulary to find the right classes to annotate. A naïve approach is to ask a separate yes/no question for each class in a given vocabulary. Such a protocol is rooted on the vocabulary, not the image content. It scales linearly in the size of the vocabulary, even when only few of the classes are present in the image (which is the typical case). Thus, it becomes very inefficient when the vocabulary is large. Let us take the ILSVRC dataset as an example: getting labels for the 200 object classes in its vocabulary would take close to 6 minutes per image (Krishna et al. 2016), despite each image containing only 1.6 classes on average. Previous methods have attempted to improve on this by using a hierarchical representation of the class vocabulary to quickly reject certain groups of labels (Lin et al. 2014; Deng et al. 2014). This reduces the annotation complexity to sub-linear in the vocabulary size. But even with these sophisticated methods, object class labelling remains time consuming. Using the hierarchical method of (Deng et al. 2014) to label the 200 classes of ILSVRC still takes 3 minutes per image (Russakovsky et al. 2015b).

The COCO dataset has fewer classes (80) and was labelled using the more efficient hierarchical method of (Lin et al. 2014). Even so, it still took half a minute per image.

In total, it took 20,000 hours to annotate object class labels for the COCO dataset (Lin et al. 2014). Annotating bounding boxes for these classes (stage two) additionally takes at least 5,000 hours, even when using efficient box drawing interfaces (Papadopoulos et al. 2017a; Kuznetsova et al. 2018). Moreover, the two stages cannot be easily merged to increase efficiency, due to the complexity of hierarchical methods. Such a combined class and box labelling stage would anyhow be sequential, as annotators cannot simultaneously use the mouse and keyboard to mark the location and provide the class name.

Thus, despite the recent advances in object class label and bounding box annotation, annotating large datasets still requires tremendous amounts of time. At the same time, (Sun et al. 2017) showed that performance of current deep neural networks is not saturated. These models still benefit from more data, which motivates the community to collect and annotate even larger datasets.

In this paper we propose to use speech, together with mouse pointing, to aid the annotation of such datasets. First, we use speech for object class labelling and show that it enables significant speed gains (Sec. 3). Then, we show that speech allows to naturally combine class and box labelling into one task: annotators mark an object location via the mouse and provide its class label via speech at the same time (Sec. 4). This simultaneous *class and box labelling* allows to annotate class labels at zero additional cost, compared to annotating bounding boxes alone.

Annotating images via speaking and pointing has multiple strong advantages: (i) it leads to significant speed gains, as saying the class names is fast: people can say 150 words per minute when describing images (Vaidyanathan et al. 2018). In comparison, people normally type at 30-100 words per minute (Karat et al. 1999; Clarkson et al. 2005). (ii) speaking allows for direct access to the class name via simply *saying it*. Thereby annotators label classes of objects that they see, *i.e.* the task is rooted on the image content and naturally scales with the number of annotated objects. (iii) it does not require the experiment designer to construct a natural and intuitive hierarchy to access the class labels, as in (Lin et al. 2014; Deng et al. 2014). (iv) speaking and pointing can be done in parallel (Kahneman 1973; Oviatt 2003). This allows annotators to concurrently solve multiple tasks, such as providing the semantic label and the location of an object. In fact, people naturally choose to point for providing spatial information and to speak for semantic information when using multimodal interfaces (Oviatt 2003). (v) it makes the interface design extremely simple, which is what allows to combine object class labelling and bounding box annotation into a single task.

Using speech as an input modality, however, poses certain challenges. In order to extract object annotations from speech and mouse inputs, several technical challenges need to be tackled. These include transcribing the speech, inferring class labels, and aligning them with object location annotations (Sec. 5). Furthermore, as speech is free-form in nature, annotators need to be trained to know the class vocabulary to be annotated, in order to not label other objects or forget to annotate some classes. In Sec. 3.2 & 4.2 we address these challenges, which allows us to design annotation interfaces for fast and accurate labelling.

We validate our approach with extensive experiments (Sec. 6 & 7). In particular we:

- Show that speech enables fast object class labelling: 2.3× faster on the COCO dataset (Lin et al. 2014) than the hierarchical approach of (Lin et al. 2014), and 14.9× faster than (Deng et al. 2014) on ILSVRC (Russakovsky et al. 2015a).
- Show that the class labelling can be embedded into the bounding box annotation stage, which allows to produce class labels at zero additional cost. On

COCO, this makes the overall process 1.9× faster than the two-stage approach.
- Demonstrate that our method scales to large vocabularies.
- Show that through our training task annotators learn to use the provided vocabulary for naming objects with high fidelity.
- Analyze the accuracy of models for automatic speech recognition (ASR) and show that it supports deriving high-quality annotations from speech.

This paper is an extension of our preliminary work (Gygli and Ferrari 2019), which focused only on object class labelling. It introduces a new annotation protocol to simultaneously annotate objects with bounding boxes and class labels (Sec. 4), new experiments (Sec. 7), and a better method for temporally segmenting and aligning speech with object location annotations (Sec. 5).

2 Related Work

Using speech as an input modality has a long history (Bolt 1980) and is recently emerging as a research direction in Computer Vision (Dai 2016; Vasudevan et al. 2017; Vaidyanathan et al. 2018; Harwath et al. 2018). To the best of our knowledge, however, our paper is the first to show that speech allows for more efficient object class labelling than (Lin et al. 2014; Deng et al. 2014) and enables simultaneous class and box labelling. We now discuss previous works in the areas of leveraging speech, efficient object class labelling, learning from point supervision and bounding box annotation.

Leveraging speech inputs. To point and speak is an efficient and natural way of human communication. Hence, this approach was quickly adopted when designing computer interfaces: as early as 1980, (Bolt 1980) investigates using speech and gestures for manipulating shapes. Most previous works in this space analyse what users choose when offered different input modalities (Hauptmann 1989; Oviatt 1996; Oviatt et al. 1997; Oviatt 2003), while only few approaches focus on the added efficiency of using speech. The most notable such work is (Pausch and Leatherby 1991), which measures the time needed to create a drawing in MacDraw. They compare using the tool as is, which involves selecting commands via the menu hierarchy, to using voice commands. They show that using speech gives an average speedup of 21% and mention this is a "lower bound", as the tool was not designed with speech in mind.

In Computer Vision, (Vasudevan et al. 2017) detect objects given spoken referring expressions, while (Harwath et al. 2018) learn an embedding from spoken image-caption pairs. Their approach obtains promising first results, but still performs inferior to learning on top of textual captions produced by Google's automatic speech recognition. (Damen et al. 2018) annotates the EPIC-KITCHENS dataset based on spoken free-form narratives, which cover some of the objects present in the image. These narratives are however transcribed *manually*, and then object class labels are derived from transcribed nouns, again manually. Instead, our approach is fully automatic and we exhaustively label all objects from a given vocabulary. Finally, more closely related to our work, (Vaidyanathan et al. 2018) re-annotated a subset of COCO with spoken scene descriptions and human gaze. While efficient, free-form scene descriptions are noisier when used for object class labelling, as annotators might refer to objects with ambiguous names, mention nouns that do not correspond to objects shown in the image (Vaidyanathan et al. 2018), or there might be inconsistencies in naming the same object classes across different annotators. Our approach avoids the additional complexities of parsing free-form sentences to extract object names and gaze data to extract object locations.

Efficient object class labelling. The naïve approach to annotating the presence of object classes grows linearly with the size of the vocabulary (one binary present / absent question per class). The idea behind sub-linear schemes is to group the classes into meaningful super-classes, such that several of them can be ruled out at once. If a super-class (*e.g.* animals) is not present in the image, then one can skip the questions for all its subclasses (cat, dog, *etc.*). This grouping of classes can have multiple levels. The annotation schemes behind COCO (Lin et al. 2014) and ILSVRC (Deng et al. 2014; Russakovsky et al. 2015a) datasets both fall into this category, but they differ in how they define and use the hierarchy.

ILSVRC (Russakovsky et al. 2015a) was annotated using a series of hierarchical questions (Deng et al. 2014). For each image, 17 top-level questions were asked (*e.g.* "Is there a living organism?"). For groups that are present, more specific questions are asked subsequently, such as "Is there a mammal?", "Is there a dog?", *etc.* The sequence of questions for an image is chosen dynamically, such that the they allow to eliminate the maximal number of labels at each step (Deng et al. 2014). This approach, however, involves repeated visual search, in contrast to ours, which is guided by the annotator scanning the image for objects, done only once. Overall, this scheme takes close to 3 minutes per image (Russakovsky et al. 2015b) for annotating the 200 classes of ILSVRC. On top of that, constructing such a

hierarchy is not trivial and influences the final results (Russakovsky et al. 2015a).

In the protocol used to create COCO (Lin et al. 2014), annotators are asked to mark one object for each class present in an image by choosing its symbol from a two-level hierarchy and dragging it onto the object (Fig. 7). While this allows to take the image, rather than the questions as the root of the labelling task, it requires repeatedly searching for the right class in the hierarchy, which induces significant time cost. In our interface, such an explicit class search is not needed, which speeds up the annotation process.

Rather than using a hierarchy, (Kuznetsova et al. 2018) annotated object class labels on the Open Images dataset by relying on an image classifier. The classifier creates a shortlist of object classes likely to be present, which are then verified by annotators using binary questions. The shortlist is generated using a predefined threshold on the classifier scores. Thus, this approach trades off completeness for speed. In practice, (Kuznetsova et al. 2018) asks annotators to verify 10 out of 600 classes, but report a rather low recall of 59%, despite disregarding "difficult" objects in evaluation.

Point supervision. The output of our interface for object class labelling is a list of all classes present in the image with a point on one object for each. This kind of labelling is efficient and provides useful supervision for several image (Papadopoulos et al. 2017b; Bearman et al. 2016; Laradji et al. 2018) and video (Mettes et al. 2016; Manen et al. 2017) object localization tasks. In particular, (Papadopoulos et al. 2017b; Bearman et al. 2016; Manen et al. 2017) show that for their task, point clicks deliver better models than other alternatives when given the same annotation budget.

Bounding box annotation. Typically, bounding boxes are annotated given image-level labels (Lin et al. 2014; Su et al. 2012; Russakovsky et al. 2015a). (Su et al. 2012) reports that it takes 25.5 seconds to draw a bounding box. Recently, (Papadopoulos et al. 2017a) proposed extreme clicking, where a bounding box is annotated by clicking on four extreme points of the object. Using this procedure makes bounding box annotation significantly faster: it takes only 7.4 seconds on average to draw bounding boxes around for the objects in the Open Images dataset (Kuznetsova et al. 2018). Hence, we also use extreme clicking in our simultaneous class and box labelling.

3 Object Class Labelling

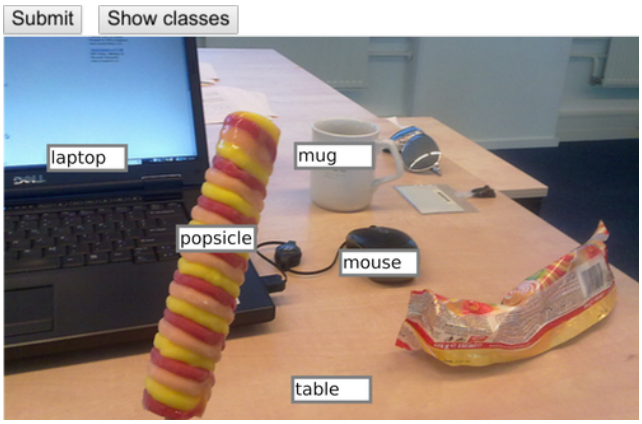
We now describe our interface for using speech in the first annotation stage: determining the presence or absence of object classes in an image (Sec. 3.1). Before annotators can proceed to the main task, we require them to pass a training stage. This helps them memorise the class vocabulary and get confident with using the interface (Sec. 3.2). From the output of this annotation task we derive class labels by transcribing the recorded speech and mapping it to class names as described in Sec. 5.1.

Fig. 2: **Our interface for object class labelling.** Given an image the annotator is asked to click on one object per class and say its name. To aid memory, we additionally allow to review the class vocabulary through the "Show classes" button.

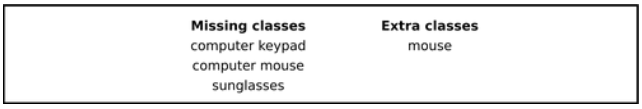
3.1 Annotation task

First, annotators are presented with the class vocabulary and instructed to memorise it. Then, they are asked to label images with object classes from the vocabulary, by scanning the image and saying the names of the different classes they see. Hence, this is a simple visual search task that does not require any context switching. While we are primarily interested in object class labels, we ask annotators to click on one object for each class, as the task naturally involves finding objects anyway. This matches the COCO protocol, allowing for direct comparisons (Sec. 6.1). It further provides information that can be used as input to weakly-supervised methods (Bearman et al. 2016; Papadopoulos et al. 2017b). Fig. 2 shows the interface with an example image.

To help annotators restrict the labels they provide to the predefined vocabulary, we allow them to review it using a button that shows all class names including their symbols.



(a)



(b)

Fig. 3: **Training process for object class labelling**. 3a shows the training task: marking an object per class with a click and saying and writing its name. 3b shows the feedback given after each image.

3.2 Annotator training

Before tackling the main task, annotators go through a training stage which gives feedback after every image and also aggregated statistics after 80 images. If they meet our accuracy targets, they can proceed to the main task. If they fail, they can repeat the training until they succeed.

Purpose of training. Training helps annotators to get confident with an interface and allows to ensure they correctly solve the task and produce high-quality labels. As a consequence, it has become common practice (Russakovsky et al. 2015a; Su et al. 2012; Lin et al. 2014; Kuznetsova et al. 2018; Papadopoulos et al. 2017b).

While we want to annotate classes from a predefined vocabulary, speech is naturally free-form. In our initial experiments we found that annotators produced lower recall compared to an interface which displays an explicit list of classes, due to this discrepancy. Hence, we designed our training task to ensure annotators memorise the vocabulary and use the correct object names. Indeed, after training annotators with this process they rarely use object names that are not in the vocabulary and obtain a high recall, comparable to (Lin et al. 2014) (Sec. 6.2 & 6.4).

Training procedure. The training task is similar to the main task, but we additionally require annotators to type the words they say (Fig. 3a). This allows to measure transcription accuracy and dissect different sources of error in the final class labelling (Sec. 6.4). After each image we give immediate feedback listing their mistakes, by comparing their answers against a pre-annotated ground truth. This helps annotators memorise the class vocabulary and learn to spot all object classes (Fig. 3b). We base this feedback on the written words, rather than the transcribed audio, for technical simplicity.

Passing requirements. At the beginning of training, annotators are given targets on the minimum recall and precision they need to reach. Annotators are required to label 80 images and are given feedback after every image, listing their errors on that image, and on how well they do overall with respect to the given targets. If they meet the targets after labelling 80 images, they successfully pass training. In case of failure, they are allowed to repeat the training as many times as they want.

4 Simultaneous Bounding Box and Class Labelling

We propose an interface to simultaneously annotate bounding boxes and class labels, thus combining the two standard stages into one. Before annotators can proceed to the main task, we require them to pass a training stage (Sec. 4.2). The annotation task produces a series of bounding boxes with start and end times and an audio recording for each image. We will transform this data into the final object annotations by deriving object classes from speech and matching them to bounding boxes as described in Sec. 5.

4.1 Annotation task

As in Sec. 3.1, annotators are presented with the class vocabulary and instructed to memorise it (but can again review it later). Then, they are asked to annotate *all* objects of all classes that are in the vocabulary. For each object, the annotator simultaneously draws a bounding box while saying its class name. We annotate bounding boxes using the efficient extreme clicking method (Papadopoulos et al. 2017a), which requires clicking on the top, bottom, left- and right-most point of an object. Hence, this annotation task requires speaking and clicking, which can naturally be done in parallel (Kahneman 1973; Oviatt 2003). Fig. 4 shows the interface with example annotations.

Fig. 4: **Our interface for simultaneous class and box labelling.** Annotators are asked to mark all objects of a given vocabulary with a bounding box and their class names.

Fig. 5: **Transcription segmentation.** Our method assigns a cost to each subsequence of words. Then, it segments the transcription into the list of object class names that has minimal cost. In this example, segmenting into $\{\text{person}, \text{dining table}\}$ has zero cost, while all other segmentations have cost > 0, as other possible subsequences such as "person dining" are not in the vocabulary.

4.2 Annotator training

We train annotators for the task of drawing bounding boxes and saying object class names. Thereby we use two training steps. The first trains annotators to quickly draw accurate bounding boxes of a given object class, following the training protocol of (Papadopoulos et al. 2017a; Kuznetsova et al. 2018). The second step trains annotators to say the correct class names. It is similar to the training procedure for object class labelling (Sec. 3.2), except that annotators have to mark an objects location with a bounding box, rather than just a single click.

5 Transcription and temporal alignment

The interfaces presented in Sec. 3 & Sec. 4 both output an audio recording and a sequence of object locations $(o_1, o_2, \ldots)$, each with a time interval. The object location is a single click in Sec. 3 and a bounding box in Sec. 4. To produce the final object annotations, we first transcribe and segment the audio to derive a sequence of object class labels (Sec. 5.1). Then, we temporally align the class labels to object locations using global sequence alignment (Sec. 5.2). The same alignment algorithm applies generally to both Sec. 3 & 4.

5.1 Infer class names from speech

We transcribe the audio with Google's automatic speech recognition API[1]. The API outputs a sequence of utterances (continuous speech blocks, enclosed by pauses) with a ranked list of transcription alternatives for each. We first segment the different transcriptions into class names and then choose the most likely transcription for each utterance.

Segmenting a transcription. While the transcription for an utterance would ideally only contain a single class name, we find that in practice it sometimes contains two or more classes, *e.g.* "person dining table". This happens when the pause between saying different class names is too short. Thus, we propose an algorithm to automatically segment the transcription into the most likely sequence of class names.

We assign a cost to each subsequence of words, *e.g.* "person", "person dining", *etc.* (Fig. 5). The cost corresponds to the dissimilarity between the subsequence and the nearest class name in the vocabulary. Thereby we represent a subsequence with its word2vec (Mikolov et al. 2013) vector and use the cosine distance between word vectors as a dissimilarity measure. For subsequences that are in the vocabulary, *e.g.* "person" and "dining table", the cost is thus zero. For subsequences that are semantically similar to words in the vocabulary, *e.g.* "table" is similar to "dining table", the cost is above zero, but still low (0.2 in the figure). For subsequences that do not even correspond to a valid noun, *e.g.* "person dining", we assign a fixed high cost (1 in the figure).

The total cost $\epsilon(s)$ of a segmentation s is the sum over the cost of each subsequence in s. By relying on word2vec distance, we assign low costs to synonyms of vocabulary words. *e.g.* a segmentation of "person sofa" into $\{\text{person}, \text{sofa}\}$ would correctly get a low total cost even when the vocabulary contains "couch" instead of "sofa".

[1] `https://cloud.google.com/speech-to-text/`

We find the most likely segmentation s^* of a transcription as the one with the lowest total cost using dynamic programming. In the example from Fig. 5, this would be $s^* = \{\text{person}, \text{dining table}\}$, which has segmentation cost $\epsilon(s^*) = 0$.

Selecting the most likely transcription. An utterance comes with multiple possible transcriptions, *e.g.* "person dining table","ocean dining table", "person dying table". We re-rank these transcriptions by their minimum segmentation cost (see above) and choose the one with the lowest cost. If two transcriptions have the same cost, we choose the one which the speech recognition API ranked higher. In the above example, this would allow to correctly identify "person dining table" as the correct transcription, as its most likely segmentation $\{\text{person}, \text{dining table}\}$ contains only class names from the vocabulary (and thus has zero cost). The other alternatives have a non-zero cost, as "ocean" and "dying table" are not part of the vocabulary.

In rare cases the top transcription contains an object name that is not in the vocabulary. In these cases we map the object name to the closest class in the vocabulary, using the cosine distance of their word2vec (Mikolov et al. 2013) representation.

The final output of this algorithm is a sequence of class labels $(c_1, c_2, \ldots)$, each with an associated start and end time (we can do this as the speech recognition API outputs the start and end time of each word).

5.2 Aligning class labels and object locations

We propose a method for temporally aligning the sequence of class labels $(c_1, c_2, \ldots)$ with object locations $(o_1, o_2, \ldots)$. As annotators sometimes speak before or after annotating the object location, rather than during it, aligning the two is not trivial.

Hence, we align the elements in the two sequences based on how much they temporally overlap (Fig. 6). The cost of an alignment is the sum of aligning individual elements, plus a gap penalty for each element that has no correspondence. Gaps can happen in practice, *e.g.* if an annotator discards a bounding box annotation, which would lead to a class label with no correspondence. We define the cost of aligning two elements as $\rho(c_i, o_j) = 1 - \mathrm{i}(c_i, o_j)/\mathrm{d}(c_i)$, where $\mathrm{i}(c_i, o_j)$ is the temporal overlap of the object location and class label annotation times. $\mathrm{d}(c_i)$ is the duration of saying class label c_i. The cost thus encourages aligning elements where the class name was said during the time interval at which the object location was drawn. But it does so smoothly, allowing for deviations from the ideal case.

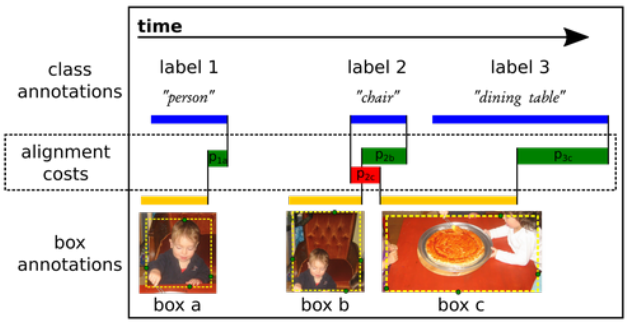


Fig. 6: **Temporal alignment.** Object locations and class labels are aligned using the time intervals during which they are annotated. The cost of an alignment depends on the temporal overlap of saying the class name and providing the object location. We find a globally optimal alignment with the Needleman-Wunsch algorithm (Needleman and Wunsch 1970). In above example, the correct alignment of label 2 with box b has higher cost than aligning it with box c (when considering this on its own). Using a global alignment technique allows to correctly align them nonetheless.

We find the optimal global alignment by relying on the Needleman-Wunsch algorithm (Needleman and Wunsch 1970). Thereby each class label c_i and object location o_j can be matched at most once and gaps are possible. The algorithm uses dynamic programming to find the best global alignment, which is the one that has the minimum cost. We found a large range of gap penalties to work well and empirically set it to 0.5.

This alignment algorithm outputs a sequence of object annotations, each consisting of a class label c and an object location o.

6 Experiments on Object Class Labelling

Here we present experiments on annotating object class labels using our speech-based interface and the hierarchical interface of (Lin et al. 2014). First, in Sec. 6.1 we re-implement the interface of (Lin et al. 2014) and compare it to the official reported results in (Lin et al. 2014). Then, we compare the two interfaces on the COCO dataset, where the vocabulary has 80 classes (Sec. 6.2). In Sec. 6.3 we scale up annotation to a vocabulary of 200 classes by experimenting on the ILSVRC dataset. Finally, Sec. 6.4 provides additional analysis such as the transcription and click accuracy as well as response times per object.

Fig. 7: Our re-implementation of the hierarchical interface of (Lin et al. 2014) for object class labelling.

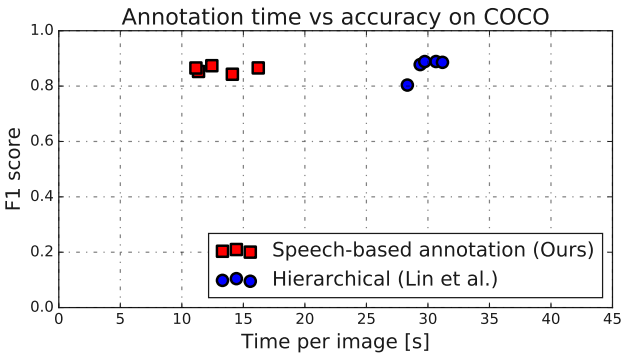


Fig. 8: Our approach *vs.* the hierarchical interface of (Lin et al. 2014). Each point in the plot corresponds to an individual annotator. F1 score is the harmonic mean between recall and precision. Dataset: COCO.

6.1 Hierarchical interface of (Lin et al. 2014)

In the interface of (Lin et al. 2014), annotators mark one object for each class present in an image by choosing its symbol from a two-level hierarchy and dragging it onto the object. While (Lin et al. 2014) reports coarse timings, we opted to re-implement their interface for fair comparison and to do a detailed analysis on how annotation time is spent (Fig. 7). First, we made five crowd workers pass a training task equivalent to that used for our interface (Sec. 3.2). Then, they annotated a random subset of 300 images of the COCO validation set (each image was annotated by all workers).

Results. Annotators take 29.9 seconds per image on average, well in line with the 27.4 seconds reported in (Lin et al. 2014). Hence, we can conclude that our implementation is equivalent in terms of efficiency.

Annotators have produced labels with 89.3% precision and 84.7% recall against the ground truth (Tab. 1). Thus, they are accurate in the labels they produce and recover most object classes. We also note that the COCO ground truth itself is not free of errors, hence limiting the maximal achievable performance. Indeed, our recall and precision are comparable to the numbers reported in (Lin et al. 2014).

Time allocation. In order to better understand how annotation time is spent, we recorded mouse and keyboard events. This allows us to estimate the time spent on searching for the right object class in the hierarchy of symbols and measure the time spent dragging the symbol. On average, search time is 14.8s and drag time 3.4s per image. Combined, these two amount to 61% of the total annotation time, while the rest is spent on other tasks such as visual search. This provides a target on the time that can be saved by avoiding these two operations, as done in our interface. In the remainder of this section, we compare our speech-based approach against this annotation method.

6.2 Our interface on COCO

We evaluate our approach and compare it to (Lin et al. 2014). Annotations with our interface were done by a new set of crowd workers, to avoid bias arising from having used the hierarchical interface before. The workers are all Indian nationals and speak English with an Indian accent. Hence, we use a model of Indian English for the automatic speech recognition. We also provide the class vocabulary as phrase hints[2], which is crucial for achieving high transcription accuracy (Sec. 6.4).

Speed and semantic accuracy. Fig. 8 and Tab. 1 show results. Our method achieves a speed-up of 2.3× over (Lin et al. 2014) at similar F1 scores (harmonic mean of precision and recall). In Sec. 6.1 we estimated that annotation could be sped up by up to 2.6× by avoiding symbol search and dragging. Interestingly, our interface achieves a speedup close to this target, confirming its high efficiency.

Despite the additional challenges of handling speech, average precision is only 1.7% lower than for (Lin et al. 2014). Hence, automatic speech transcription does not affect label quality much (we study this further in Sec. 6.4). Recall is almost identical (0.5% lower), confirming that,

[2] `https://cloud.google.com/speech-to-text/docs/basics#phrase-hints`

thanks our training task (Sec. 3.2), annotators remember what classes they have to label.

Location accuracy. We further evaluate the location accuracy of the clicks by using the ground-truth segmentation masks of COCO. Specifically, given an object annotation with class c_i, we evaluate whether its click position lies on a ground-truth segment of class c_i. If class c_i is not present in the image at all, we ignore that click in the evaluation to avoid confounding semantic and location errors.

This analysis shows that our interface leads to high location accuracy: 96.1% of the clicks lie on the object. For the hierarchical interface it is considerably lower at 90.7%. While this may seem surprising, it can be explained by the differences in the way the location is marked. In our interface one directly *clicks* on the object, while (Lin et al. 2014) requires *dragging* a relatively large, semi-transparent class symbol onto it (Fig. 7).

Parts of the speed gains of our interface are due to concurrently providing semantic and location information. However, this could potentially have a negative effect on click accuracy. To test this, we compare to the click accuracy that the annotators in (Bearman et al. 2016) obtained on the PASCAL VOC dataset. Their clicks have a location accuracy of 96.7% comparable to our 96.1%, despite the simpler dataset with larger objects on average, compared to COCO. Hence, we can conclude that clicking while speaking does not negatively affect location accuracy.

	Speech	**(Lin et al. 2014)**	**(Deng et al. 2014)**
COCO			
Recall	84.2 %	84.7 %	
Precision	87.6 %	89.3 %	
Time / image	13.1s	29.9s	
Time / label	4.5s	11.5s	
ILSVRC			
Recall	83.2 %	88.6 %	
Precision	80.3 %	76.6 %	
Time / image	12.0 sec.	31.1 sec.	≈ 179 sec.
Time / label	7.5 sec.	18.4 sec.	≈ 110 sec.

Table 1: Accuracy and speed of our interface (Speech) and hierarchical approaches (Lin et al. 2014; Deng et al. 2014) for object class labelling. Our interface is significantly faster at comparable label quality. Timings for (Deng et al. 2014) are taken from (Russakovsky et al. 2015b). Note that the numbers for Speech differ slightly from those reported in (Gygli and Ferrari 2019), due to the changes in the temporal segmentation and alignment (Sec. 5).

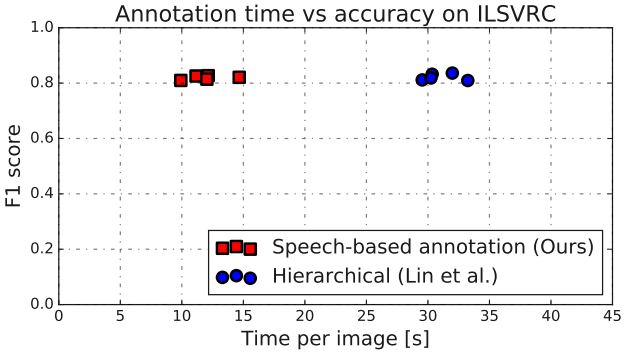


Fig. 9: Our approach *vs.* the hierarchical interface (Lin et al. 2014). Each point in the plot corresponds to an individual annotator. Dataset: ILSVRC.

6.3 Our interface on ILSVRC 2014

Here we apply our interface and the hierarchical interface of (Lin et al. 2014) to a larger vocabulary of 200 classes, using 300 images from the validation set of ILSVRC (Russakovsky et al. 2015a). For (Lin et al. 2014) we manually constructed a two-level hierarchy of symbols, based on the multiple hierarchies supplied by (Russakovsky et al. 2015a). The hierarchy consists of 23 top-level classes, such as "fruit" and "furniture", each containing between 5 to 16 object classes.

Speed and semantic accuracy. Fig. 9 shows a comparison to (Lin et al. 2014) in terms of speed and accuracy, while Fig. 13 shows example annotations produced with our interface. In Tab. 1, we also compare to the speed of (Deng et al. 2014), the method that was used to annotate this dataset. Our approach is substantially faster than both: 2.6× faster than (Lin et al. 2014) and 14.9× faster than (Deng et al. 2014). We also note that (Deng et al. 2014) only produces a list of classes present in an image, while our interface and (Lin et al. 2014) additionally yield the location of one object per class.

Despite the increased difficulty of annotating this dataset, which has considerably more classes than COCO, annotators produce high-quality labels with our interface. The F1 score is similar to that of (Lin et al. 2014) (81.7% *vs.* 82.2%). While recall is lower for our interface, precision is higher.

Fig. 10 shows a histogram of the annotation time per image. Most images are annotated extremely fast, despite the large vocabulary, as most images in this dataset contain few classes. Indeed, there is a strong correlation between the number of object classes present in an image and its annotation time (rank correlation 0.55). This highlights the advantage of methods that are rooted on the image content, rather than the vo-

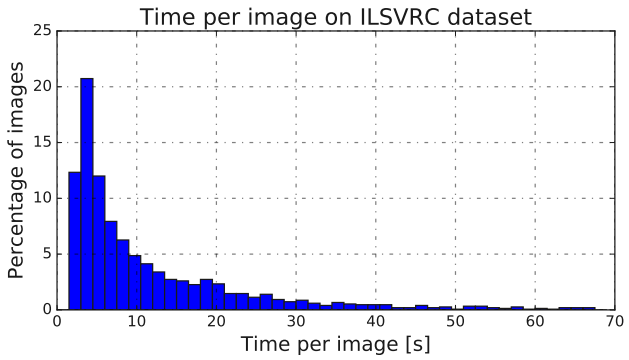


Fig. 10: Histogram of the time required to annotate an image using our interface. Dataset: ILSVRC.

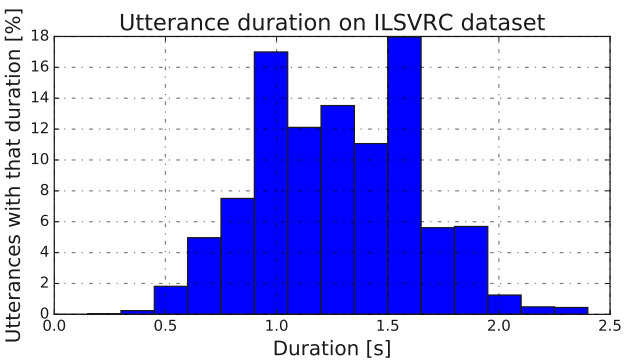


Fig. 11: Histogram of the time spent *saying* the object name on ILSVRC. Saying the object names is fast and usually takes less than 2 seconds.

cabulary: their annotation time is low for images with few classes. Instead, methods rooted on the vocabulary cannot exploit this class sparsity to a full extent. The naïve approach of asking one yes/no question per class is actually even slower the fewer objects are present, as determining the absence of a class is slower than confirming its presence (Ehinger et al. 2009).

6.4 Additional analysis of our interface

Time allocation. To understand how much of the annotation time is spent on what, we analyse timings for speaking and moving the mouse on the ILSVRC dataset. Of the total annotation time, 26.7% is spent on speaking. The mouse is moving 74.0% of the total annotation time, and 62.4% of the time during speaking. The rather high percentage of time the mouse moves during speaking confirms that humans can naturally carry out visual processing and speaking concurrently.

In order to help annotators label the correct classes, we allowed them to consult the class vocabulary, through a button on the interface (Fig. 2). This takes 7.2% of the total annotation time, a rather small share. Annotators consult the vocabulary in fewer than 20% of the images. When they consulted it, they spent 7.8 seconds looking at it, on average. Overall, this shows the annotators feel confident about the class vocabulary and confirms that our annotator training stage is effective.

In addition, we analyse the time it takes annotators to say an object name in Fig. 11, which shows a histogram of speech durations. As can be seen, most names are spoken in 0.5 to 2 seconds.

Per-click response time. In Fig. 12 we analyse the time taken to annotate the first and subsequent classes of an image in the COCO dataset. It takes 3.3s to make the first click on an object, while the second takes 2.0s only. This effect was also observed by (Bearman et al. 2016). Clicking on the first object incurs the cost of the initial visual search across the whole scene, while the second is a continuation of this search and thus cheaper (Watson and Inglis 2007; Rayner 2009; Lleras et al. 2005). After the second class, finding more classes becomes increasingly time-consuming again, as large and salient object classes are already annotated. Indeed, we find that larger objects are typically annotated first: object size has a high median rank correlation with the annotation order (-0.80). Interestingly, on the interface of (Lin et al. 2014), this effect is less pronounced (-0.50), as the annotation order is affected by the symbol search and grouping of classes in the hierarchy. Finally, our analysis shows that the annotators spend 3.9s between saying the last class name and submitting the task, indicating that they do a thorough final scan of the image to ensure they do not miss any class.

Mouse path length. To better understand the amount of work required to annotate an image we also analyse the mean length of the mouse path. We find that on ILSVRC annotators using (Lin et al. 2014) move the mouse for a 3.0× greater length than annotators using our interface. Thus, our interface is not only faster in terms of time, but is also more efficient in terms of mouse movements. The reason is that the hierarchical interface requires moving the mouse back and forth between the image and the class hierarchy (Fig. 14). The shorter mouse path indicates the simplicity and improved ease of use of our interface.

Training time. Training annotators to achieve good performance on the 200 classes of ILSVRC takes 1.6 hours for our interface, or 1 hour with the hierarchical interface of (Lin et al. 2014). Instead, annotating the full ILSVRC dataset would take 1,726 hours with our

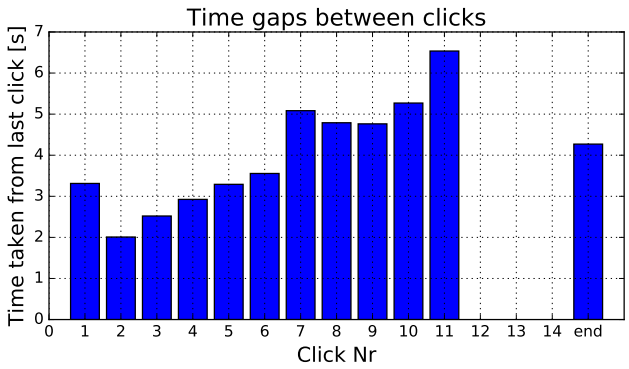


Fig. 12: Analysis of the time it takes for the first and subsequent clicks when annotating object classes on the COCO dataset.

interface *vs.* 4,474 hours with (Lin et al. 2014). Hence, the cost of training is negligible and our interface is far more efficient than (Lin et al. 2014) even after taking training into account.

Transcription accuracy. The annotator training task produces spoken and written class names for each annotated object (Sec. 3.2). Using this data we evaluate the accuracy of the automatic speech recognition (ASR). For this we only take objects into account if they have transcriptions results attached. This keeps the analysis focused on transcription accuracy by ignoring other sources of errors, such as incorrect temporal alignment or annotators simply forgetting to say the class name after they click on an object.

Tab. 2 shows the transcription accuracy in two setups: with and without using the vocabulary as phrase hints. Phrase hints allow to indicate phrases or words that are likely to be present in the speech and thus help the ASR model transcribe them correctly more often. Using phrase hints is necessary to obtain high transcription accuracy. Thanks to them, Recall@3 is at 96.5% on COCO and 97.5% on ILSVRC. Hence, the top three transcriptions usually contain the correct class name, which we then extract as described in Sec. 5.1.

In fact, we actually consider the above numbers to be a lower bound on the transcription accuracy in the main task, as here we compare the transcriptions against the raw written class names, which contain a few spelling mistakes. Moreover, here the annotators are in the training phase and hence still learning about the task. Overall, the above evidence shows that ASR achieves high accuracy, definitely good enough for labelling object class names.

Vocabulary usage. As speech is naturally free-form, we are interested in knowing how often annotators use

	Recall@1	**Recall@3**
COCO w/ hints	93.1 %	96.5 %
COCO w/o hints	70.5 %	84.7 %
ILSVRC w/ hints	93.3 %	97.5 %
ILSVRC w/o hints	70.2 %	89.5 %

Table 2: **Transcription accuracy.** Accuracy is high when using phrase hints (see text).

object names that are outside of the vocabulary. Thus, we analyse how often the written class name in the annotator training task does not match a vocabulary name. We find that on COCO annotators are essentially only using names from the vocabulary (99.5% of the cases). On ILSVRC they still mostly use names from the vocabulary, despite the greater number of classes which induces a greater risk of misremembering their names (96.3% are in vocabulary).

Some of the out-of-vocabulary names are in fact variations of names in the vocabulary. These cases can be mapped to their correct name in the vocabulary as described in Sec. 5.1. For example, for the ILSVRC dataset some annotators say "oven", which gets correctly mapped to "stove", and "traffic signal" to "traffic light". In other cases the annotators use out-of-vocabulary names because they actually label object classes that are not in the vocabulary (*e.g.* "fork" and "rat", which are not classes of ILSVRC).

We find that our annotator training task helps reducing the use of out-of-vocabulary names: on ILSVRC the use of vocabulary names increases from 96.3% in training to 97.5% in the main task.

Error analysis. To better understand the limits of our method we conducted a detailed analysis of the errors annotators make. We analyzed the recall per class and as a function of the number of classes in an image.

In terms of average recall per class we find no significant difference between the annotations produced by the hierarchical interface of (Lin et al. 2014) and our method on the COCO dataset (82.1% vs 82.8%). On the ILSVRC dataset, our method has a somewhat lower average class recall of 76.7% compared to (Lin et al. 2014) with 84.5%. This is consistent with the difference in the overall recall (Tab. 1). We attribute this difference to the increased challenge of spotting 200 different classes, without being explicitly asked about their presence.

Secondly, we analyzed recall as a function of the number of distinct object classes present in an image. On the COCO dataset, our method delivers equal or slightly better recall to the hierarchical approach for images with up to 3 classes (which is the most common case, Fig. 15). For cluttered images with more than 3 classes, the recall of our method decreases to slightly

Fig. 13: Example annotations on ILSVRC. For each click we show the three alternatives from the ASR model (orange) and the final class label (green). The first three images show typical annotations produced by our method. The last one shows a failure case: while the correct name is among the alternatives, an incorrect transcription matching a class name ranks higher, hence the final class label is wrong.

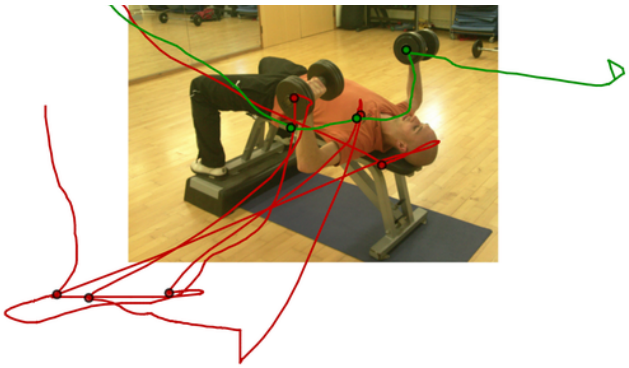


Fig. 14: A comparison of typical mouse paths produced when annotating an image with our interface (green) or with (Lin et al. 2014) (red). Circles indicate clicks. Mouse paths for our interface are extremely short, thanks to its simplicity and naturalness.

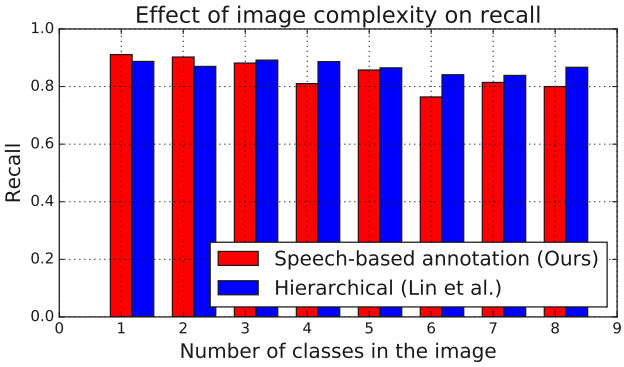


Fig. 15: **Recall as a function of the number of distinct object classes present in an image on the COCO dataset**. We find that for cluttered images, annotators tend to miss fewer objects with (Lin et al. 2014) compared to our method.

below that of the hierarchical approach. In such images objects are often small and hard to spot, hence explicitly querying for then, as done in (Lin et al. 2014), can help to find such objects. On ILSVRC, which consists of simpler images with fewer distinct classes in an image (1.6 in average), this effect is less visible.

7 Experiments on Bounding Box Annotation

We now present results on using speech to simultaneously annotate objects with a bounding box and their class label. Thereby we compare our approach (Sec. 4) to a standard two-stage approach. As in the previous experiment we use 5 crowd workers, which each annotate the 80 classes of the COCO dataset on 300 images.

Below we start by briefly explaining the two-stage baseline (Sec. 7.1), before presenting the results of our method (Sec. 7.2). Finally, Sec. 7.3 provides additional analysis of our interface.

7.1 Two-stage approach

We evaluate the standard way to annotate images with object bounding boxes (Russakovsky et al. 2015a; Kuznetsova et al. 2018; Su et al. 2012) or outlines (Lin et al. 2014), which is typically done in two stages. In the first stage, annotators are asked to mark the presence or absence of object classes in each image. For this, we use the results produced by the hierarchical interface of (Lin et al. 2014) (Sec. 6.1). Thereby we use the object class labels produced by a single annotator, randomly chosen for each image. In the second stage, annotators are given one of these class labels and are asked to draw bounding boxes for all objects of that class. For this, we use the efficient extreme clicking interface (Papadopoulos et al. 2017a). This task is repeated for each class marked as present. We use the same interface as the one presented in Sec. 4.1, adapted to this task.

Results. Tab. 3 shows results. The first stage of the two-stage approach takes 29.9 seconds per image (Sec. 6.1). Then, the second stage takes 7.4 seconds per box (Pa-

padopoulos et al. 2017a; Kuznetsova et al. 2018). We can estimate the total cost per box by diving the cost of the first stage by the average number of boxes per image (5.1s), and then adding the cost of the second stage (7.4s). This gives a total cost of 12.5 seconds per box.

Furthermore, as a sanity check we evaluate if the bounding boxes produced in this experiment are semantically correct, by comparing their class labels against the ground truth. Specifically, for each annotated bounding box, we find the ground truth box with the highest overlap and check whether the two boxes have the same label. We ignore annotated boxes for which there is no corresponding ground truth box. We find that the resulting boxes have high semantic accuracy, with 96.4% of the classes being correct. We further evaluate geometrical accuracy using mean intersection-over-union (IoU). The bounding boxes have a mean IoU of 84.4%, close to the human agreement upper-bound of 88% (Papadopoulos et al. 2017a; Kuznetsova et al. 2018). Hence, we conclude that the data produced by this baseline experiment is of high quality.

7.2 Results for Simultaneous Class and Box Labelling

We analyze the time per box for our method in Tab. 3. Simultaneously annotating one object with a bounding box and class label with our method takes 6.5 seconds on average. It thus provides a significant speedup of 1.9× over the two-stage approach.

This experiment shows the power of speech as annotation modality, as the class label can in fact be annotated at *zero additional cost* over just annotating bounding boxes. This is because speaking and pointing can be done in parallel (Kahneman 1973; Oviatt 2003). Interestingly, we find the joint approach to be even slightly faster than bounding box annotation alone. This may seem surprising, but it can be explained by how the objects are annotated in the two cases: In our approach, all classes are annotated at once. Hence, the annotator parses the image only once, actively searching for all objects across all classes in the vocabulary. In the two-stage approach instead, annotators draw bounding boxes of each class separately. For each class the image is presented again, hence requiring repeated visual search. This small extra cost translates into about $0.9s$ per box on average. While it is not an intrinsic advantage of using speech, as the two-stage approach could be reorganized with a smarter interface that asks to box all classes at once, we believe it is an interesting effect.

In terms of quality, we find that our approach produces bounding boxes with semantically accurate class labels (94.2%). This is, however, slightly below the two-stage approach (−2.2%), which can be attributed to transcription and alignment errors. Geometrical accuracy is high and similar to the two-stage approach (83.4% *vs.* 84.4%). Thus, we conclude that our method produces annotations with comparable accuracy, but at a significant speed gain of 1.9×. We show example annotations generated with our interface in Fig. 16.

	Ours (Box & Speak)	**Two-stage approach**
Semantic accuracy	94.2%	96.4%
IoU	83.4%	84.4%
Time / box	6.5 sec.	12.5 sec. (5.1s + 7.4s)

Table 3: **Bounding Box annotation results.** Our method of jointly providing class labels and boxes is 1.9× faster than the standard two-stage approach. Dataset: COCO.

7.3 Additional analysis of our interface

Concurrency of speaking and clicking. In Fig. 17 we analyze the relative time at which annotators provide bounding boxes and say the class name. We find that annotators typically do both at roughly the same time, but have a tendency to start speaking before clicking on the object. This matches previous studies, which found that annotators show multiple patterns of multimodal annotation, where one input often partially precedes the other (Oviatt et al. 1997; Oviatt 2003). In fact, we observed this effect despite instructing annotators to "mark the object and speak at the same time". We conjecture that annotators start speaking after spotting an object and while they move the mouse to the first click position. In fact this tendency is occasionally so strong that there is no temporal overlap between saying the class name and marking the object location (3.6% of the cases).

As a consequence of this variability, correctly aligning bounding boxes and class names is not trivial. One potential solution to this challenge could be to assume a fixed pattern and force annotators to use it. As an example: we could only transcribe the speech between the first and last click of annotating a bounding box. However, this will lead to deriving the class name from incomplete audio when the annotator does not strictly follow this imposed pattern. Hence, we opt to let the annotators behave naturally, without enforcing a fixed annotation pattern. Instead we handle the resulting differences between the time they speak and point via our

Fig. 16: **Example annotations produced with simultaneous class and box labelling on COCO.** Original annotations in **red**, annotations produced with our method in **green**. Our method produces accurate bounding boxes and semantic labels. In some cases, annotators delimit object classes differently than in the original annotation (center). Annotators are able to simultaneously annotate semantic labels and boxes, even for complex images with many objects present (right, original annotations omitted for better visibility).

robust alignment method (Sec. 5) which we evaluate next.

Comparison of alignment methods. While our previous paper (Gygli and Ferrari 2019) used a heuristic method for temporal segmentation and alignment, this work proposes a principled approach to align the class labels and object location annotations (Sec. 5). Using our method leads to class labels with a semantic accuracy of 94.2%, compared to 80.9% for (Gygli and Ferrari 2019). We also find that some boxes have no class label attached, due to issues in speech recognition, alignment errors or annotators forgetting to say the class name. For our alignment method, 1.2% of the bounding boxes have no label, compared to 10.8% for (Gygli and Ferrari 2019). Hence, our method is not only more principled, it also minimizes alignment errors, which leads to considerably better accuracy in practice. At the same time it remains fast. Inferring class names and aligning them with the bounding box annotations takes $\approx 0.025s$ per image on average, on both ILSVRC and COCO.

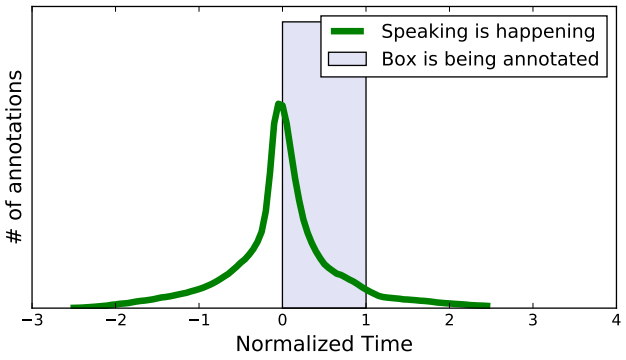


Fig. 17: **Synchronicity of speaking and box drawing.** The green curve shows the number of objects (vertical axis) for which the annotator was speaking at a particular point in time (horizontal axis). The horizontal axis is normalized by the amount of time needed to draw the box for a particular object. This curve shows that people typically start speaking a little before the do the first click on the object (time point at 0) and finish speaking mostly by the time they did the last click on the object (time point at 1).

8 Discussion

Based on our experiments and observations we now discuss some of the insights on the advantages, limitations and open questions in using speech-based annotation.

Annotating irrelevant images. In order to be able to measure semantic and location accuracy of our method, we re-annotated existing datasets. Thus, the images we used all contain at least one object of the classes contained in the vocabulary. We did not evaluate how our method compares to previous methods on images that do not contain contain any objects of the relevant classes. However, this case is rare in practice, as images in standard datasets are not uniformly sampled out of the space of all possible images. Instead, datasets are typically created by defining a fixed vocabulary of classes and then explicitly retrieving images that contain these classes from web search engines, *e.g.* COCO (Lin et al. 2014) and ILSVRC (Russakovsky et al. 2015a). More recently, the Open Images dataset was constructed by uniformly sampling images from Flickr, but then annotates a large vocabulary (600 classes) deliberately chosen to cover objects that are frequent and important in these images (Kuznetsova et al. 2018). Hence, essentially all images contain at least one object from the vocabulary. Besides, we believe that even on irrelevant images our method would

not be slower than hierarchical approaches (Lin et al. 2014; Deng et al. 2014). In our method, the annotator would spend their time carefully scanning the whole image looking for any class in the vocabulary. In the hierarchical methods instead, the annotator would have to scan the image looking for any class of each top-level supercategory in turn (e.g. "living organisms", "carpentry items" or "items that run on electricity" in ILSVRC). While the latter might be a simpler task, it needs to be repeated for each supercategory.

Scaling to larger vocabularies. We experiment with 200 classes, a size which lies in the ballpark of the largest current datasets for object detection. Scaling to larger vocabularies is a challenge for any annotation method. While our method relies on memorization of the class names for fast annotation, we enable the annotator to quickly review the class vocabulary during annotation (Sec. 3.1). In practice, annotators rarely depend on this and only 7.2% of the total annotation time is spend looking up class names for a vocabulary of 200 classes (Sec. 6.4). This suggests that our approach might scale to many more classes as well. Importantly, scaling hierarchical methods also has its challenges, most notably the construction of a hierarchical representation of classes that is intuitive for annotators to use. Even for 200 classes, this is not trivial and influences the final results as noted by (Russakovsky et al. 2015a).

Generally, scaling exhaustive annotation to thousands of classes is an unsolved problem and it is unclear how well any existing method would work in practice as no experiments have been reported (as opposed to annotating a few classes from a large vocabulary in each image (Deng et al. 2009). In our experience the main challenge is not memorizing the class names, but teaching annotators how to recognize, distinguish and delimit classes within a large vocabulary. For example, in ILSVRC, annotated with a hierarchical interface, lobsters and scorpions are often confused and sometimes annotated as both. How to handle this challenge is a topic of ongoing research (Pont-Tuset et al. 2019).

Mapping speech to object classes. In preliminary experiments we made annotators manually select the class name out of the 3 most likely classes according to our method (Sec. 5.1). However, we found that such a manual selection takes considerable time, reducing the efficiency gains of using speech. Hence, we opted for automatically selecting the most likely class (Sec. 5.1). This approach is significantly faster while still delivering accurate annotations (Tab. 1 & 2), hence providing a better speed-accuracy trade-off.

9 Conclusion

We use multimodal inputs for fast image annotation. At the core of our method lies speech: annotators provide class labels by simply by saying the names of the objects that are present in an image. We have proposed two kinds of speech-based interfaces: First, an interface for object class labelling, a task that has traditionally been time consuming and difficult to design. We have shown that our method offers considerable speed gains, thanks to speech: it is $2.3\times - 14.9\times$ faster than previous methods (Lin et al. 2014; Deng et al. 2014). Second, an interface for simultaneous class and box labelling. Previous methods annotate the two in separate stages (Russakovsky et al. 2015a; Kuznetsova et al. 2018; Su et al. 2012). Instead, we have shown that using speech allows to naturally combine them, which makes the overall process $1.9\times$ faster than previous methods. This is thanks to the fact that saying the class name while drawing a bounding box can be done at zero additional cost. Finally, we have conducted a detailed analysis of our interfaces, speech transcription and temporal alignment. We believe this offers helpful insights for building even more efficient annotations tools in the future.

References

Bearman A, Russakovsky O, Ferrari V, Fei-Fei L (2016) What's the point: Semantic segmentation with point supervision. In: ECCV 4, 9, 10

Bolt RA (1980) "Put-that-there": Voice and gesture at the graphics interface. In: SIGGRAPH 3

Clarkson E, Clawson J, Lyons K, Starner T (2005) An empirical study of typing rates on mini-qwerty keyboards. In: CHI 2

Dai D (2016) Towards cost-effective and performance-aware vision algorithms. PhD thesis, ETH Zurich 3

Damen D, Doughty H, Maria Farinella G, Fidler S, Furnari A, Kazakos E, Moltisanti D, Munro J, Perrett T, Price W, et al. (2018) Scaling Egocentric Vision: The EPIC-KITCHENS Dataset. In: ECCV 3

Deng J, Dong W, Socher R, Li LJ, Li K, Fei-fei L (2009) ImageNet: A large-scale hierarchical image database. In: CVPR 15, 17

Deng J, Russakovsky O, Krause J, Bernstein MS, Berg A, Fei-Fei L (2014) Scalable multi-label annotation. In: CHI 2, 3, 9, 15

Ehinger KA, Hidalgo-Sotelo B, Torralba A, Oliva A (2009) Modelling search for people in 900 scenes: A combined source model of eye guidance. Visual cognition 10

Gygli M, Ferrari V (2019) Fast Object Class Labelling via Speech. In: CVPR 3, 9, 14

Harwath D, Recasens A, Surís D, Chuang G, Torralba A, Glass J (2018) Jointly Discovering Visual Objects and Spoken Words from Raw Sensory Input. In: ECCV 3

Hauptmann AG (1989) Speech and gestures for graphic image manipulation. ACM SIGCHI 3

Kahneman D (1973) Attention and effort. Citeseer 2, 5, 13

Karat CM, Halverson C, Horn D, Karat J (1999) Patterns of entry and correction in large vocabulary continuous speech recognition systems. In: ACM SIGCHI, ACM 2

Krishna RA, Hata K, Chen S, Kravitz J, Shamma DA, Fei-Fei L, Bernstein MS (2016) Embracing error to enable rapid crowdsourcing. In: CHI 2

Kuznetsova A, Rom H, Alldrin N, Uijlings J, Krasin I, Pont-Tuset J, Kamali S, Popov S, Malloci M, Duerig T, Ferrari V (2018) The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. arXiv preprint arXiv:181100982 1, 2, 4, 5, 6, 12, 13, 14, 15

Laradji IH, Rostamzadeh N, Pinheiro PO, Vazquez D, Schmidt M (2018) Where are the blobs: Counting by localization with point supervision. arXiv preprint arXiv:180709856 4

Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick C (2014) Microsoft COCO: Common objects in context. In: ECCV 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 14, 15, 17

Lleras A, Rensink RA, Enns JT (2005) Rapid resumption of interrupted visual search: New insights on the interaction between vision and memory. Psychological Science 10

Manen S, Gygli M, Dai D, Van Gool L (2017) PathTrack: Fast Trajectory Annotation with Path Supervision. In: ICCV 4

Mettes P, van Gemert JC, Snoek CG (2016) Spot on: Action localization from pointly-supervised proposals. In: ECCV 4

Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781 6, 7

Needleman SB, Wunsch CD (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of molecular biology 7

Oviatt S (1996) Multimodal interfaces for dynamic interactive maps. In: ACM SIGCHI 3

Oviatt S (2003) Multimodal interfaces. The human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications 2, 3, 5, 13

Oviatt S, DeAngeli A, Kuhn K (1997) Integration and synchronization of input modes during multimodal human-computer interaction. In: CHI 3, 13

Papadopoulos DP, Uijlings JR, Keller F, Ferrari V (2017a) Extreme clicking for efficient object annotation. In: ICCV 2, 4, 5, 6, 12, 13

Papadopoulos DP, Uijlings JR, Keller F, Ferrari V (2017b) Training object class detectors with click supervision. In: CVPR 4, 5

Pausch R, Leatherby JH (1991) An empirical study: Adding voice input to a graphical editor. J American Voice Input/Output Society 3

Pont-Tuset J, Gygli M, Ferrari V (2019) Natural vocabulary emerges from free-form annotations. arXiv preprint arXiv:190601542 15

Rayner K (2009) Eye movements and attention in reading, scene perception, and visual search. Quarterly Journal of Experimental Psychology 10

Russakovsky O, Lin Y, Yu K, Fei-Fei L (2012) Object-centric spatial pooling for image classification. In: ECCV

Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg A, Fei-Fei L (2015a) ImageNet large scale visual recognition challenge. IJCV 1, 2, 3, 4, 5, 9, 12, 14, 15

Russakovsky O, Li LJ, Fei-Fei L (2015b) Best of both worlds: human-machine collaboration for object annotation. In: CVPR 2, 3, 9

Su H, Deng J, Fei-Fei L (2012) Crowdsourcing annotations for visual object detection. In: AAAI Human Computation Workshop 1, 4, 5, 12, 15

Sun C, Shrivastava A, Singh S, Gupta A (2017) Revisiting unreasonable effectiveness of data in deep learning era. In: ICCV 2

Vaidyanathan P, Prud E, Pelz JB, Alm CO (2018) SNAG : Spoken Narratives and Gaze Dataset. ACL 2, 3

Vasudevan AB, Dai D, Van Gool L (2017) Object Referring in Visual Scene with Spoken Language. In: CVPR 3

Watson DG, Inglis M (2007) Eye movements and time-based selection: Where do the eyes go in preview search? Psychonomic Bulletin & Review 10

Appendix A - Two-level Hierarchy for ILSVRC

For reference we provide the hierarchy we constructed to use the interface of (Lin et al. 2014) with the 200 class vocabulary of the ILSVRC dataset (Deng et al. 2009). The hierarchy is based on the hierarchy of questions supplied in (Deng et al. 2009), but modified to balance the size of the groups and reduced to two-levels. It consists of 22 semantic groups and a small group of "misc objects":

1. Wind instruments:
   (a) trumpet; (b) saxophone; (c) trombone; (d) flute; (e) oboe; (f) harmonica; (g) french horn; (h) accordion
2. Other musical instruments:
   (a) piano; (b) guitar; (c) violin; (d) chime; (e) maraca; (f) drum; (g) cello; (h) banjo; (i) harp
3. Fruit:
   (a) pineapple; (b) fig; (c) orange; (d) banana; (e) strawberry; (f) apple; (g) lemon; (h) pomegranate
4. Other food:
   (a) pizza; (b) guacamole; (c) popsicle; (d) hamburger; (e) hotdog; (f) burrito; (g) pretzel; (h) mushroom; (i) bagel; (j) artichoke; (k) cucumber; (l) bell pepper; (m) cabbage
5. Clothing:
   (a) miniskirt; (b) diaper; (c) brassiere; (d) bathing cap; (e) bow tie; (f) helmet; (g) tie; (h) swimming trunks; (i) swimsuit; (j) hat; (k) sunglasses
6. Flying Animals:
   (a) bee; (b) ladybug; (c) butterfly; (d) dragonfly; (e) bird
7. Felines and Canines:
   (a) tiger; (b) lion; (c) domestic cat; (d) fox; (e) dog
8. Animals with hooves:
   (a) camel; (b) hippopotamus; (c) swine; (d) cattle; (e) zebra; (f) sheep; (g) horse; (h) antelope
9. Animals with 6 or more legs:
   (a) lobster; (b) scorpion; (c) isopod; (d) centipede; (e) ant; (f) tick
10. Animals with no legs:
    (a) snake; (b) goldfish; (c) jellyfish; (d) ray; (e) snail; (f) starfish; (g) whale; (h) seal
11. Other animals:
    (a) red panda; (b) porcupine; (c) giant panda; (d) rabbit; (e) koala; (f) elephant; (g) otter; (h) squirrel; (i) monkey; (j) hamster; (k) skunk; (l) armadillo; (m) bear; (n) frog; (o) lizard; (p) turtle
12. Vehicles:
    (a) airplane; (b) golfcart; (c) watercraft; (d) train; (e) bus; (f) snowmobile; (g) bicycle; (h) unicycle; (i) snowplow; (j) car; (k) motorcycle; (l) cart
13. Cosmetics:
    (a) lipstick; (b) face powder; (c) perfume; (d) hair spray; (e) cream
14. Medical items:
    (a) neck brace; (b) stethoscope; (c) band aid; (d) syringe; (e) stretcher; (f) crutch
15. Furniture:
    (a) bench; (b) chair; (c) bookshelf; (d) babys bed; (e) table; (f) sofa; (g) filing cabinet
16. Carpentry items:
    (a) axe; (b) nail; (c) power drill; (d) chain saw; (e) screwdriver; (f) hammer
17. School supplies:
    (a) pencil box; (b) pencil sharpener; (c) rubber eraser; (d) ruler; (e) binder
18. Game equipment:
    (a) baseball; (b) golf ball; (c) tennis ball; (d) racket; (e) rugby ball; (f) volleyball; (g) ping-pong ball; (h) croquet ball; (i) basketball; (j) soccer ball; (k) puck
19. Sports equipment:
    (a) dumbbell; (b) balance beam; (c) horizontal bar; (d) ski; (e) bow; (f) punching bag
20. Consumer electronics:
    (a) remote; (b) digital clock; (c) computer mouse; (d) computer keypad; (e) laptop; (f) printer; (g) iPod; (h) screen; (i) tape player; (j) microphone
21. Electronic appliances:
    (a) washer; (b) coffee maker; (c) microwave; (d) waffle iron; (e) toaster; (f) refrigerator; (g) stove; (h) dishwasher; (i) vacuum; (j) electric fan; (k) hair drier
22. Non-electric kitchen items:
    (a) bowl; (b) ladle; (c) salt shaker; (d) can opener; (e) cocktail shaker; (f) frying pan; (g) spatula; (h) plate rack; (i) strainer; (j) corkscrew; (k) water bottle; (l) mug; (m) pitcher; (n) wine bottle; (o) milk can
23. Misc objects:
    (a) person; (b) traffic light; (c) flowerpot; (d) purse; (e) backpack; (f) plastic bag; (g) lamp; (h) beaker; (i) soap dispenser