# Supplementary Materials: Estimating the Semantic Density of Visual Media

Anonymous Author(s)

## ANNOTATION DISTRIBUTION PER DATA SOURCE

The distributions of annotations per image vary substantially across the different considered data sources. In Figure 1, we show the distribution of annotations per image, normalized by data source. We cap the axis by 20% of the collection on the y and 100 annotations on the x-axis for improved visibility, as the missing parts do not contain relevant information.

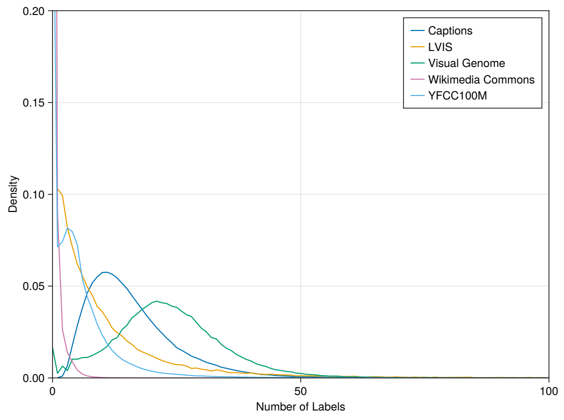

**Figure 1: Relative Distribution of the number of labels per image for each of the five data sources.**

The cumulative distributions are shown in Figure 2. Here, as well, we can see that while the data sources have a median with a comparatively small number of annotations, they all have a very long tail.

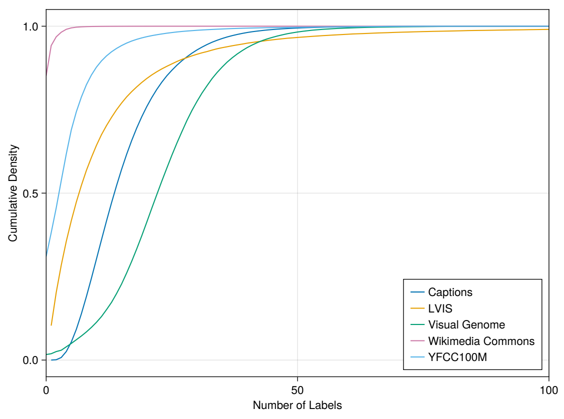

**Figure 2: Cumulative Distribution of the number of labels per image for each of the five data sources.**

## HUMAN-IN-THE-LOOP SORT

The comparison mechanism for our human-in-the-loop sort approach is outlined in pseudo-code in Listings 1 and 2. We implement the mechanism as a custom comparator (Listings 1) that can be used in the sorting mechanism provided by the standard library (Listings 2). Specifically, our implementation uses Kotlin and is available via *repository link removed for double-blind review*. In our experiment, the *requiredVotes* parameter was set to 3.

**Algorithm 1** Comparator with voting mechanism.

```
$elements \leftarrow list(\cdots)$
$votes \leftarrow emptyMap < Pair, Set < Id >> ()$
$accepted \leftarrow emptySet < Pair > ()$

function VOTE(id, o1, o2)
    $pair \leftarrow Pair(o1, o2)$
    $votes[pair].append(id)$
    if $size(votes[pair]) \geq requiredVotes$ then
        $accepted.append(pair)$
    end if
end function

function COMPARE(o1, o2)
    if $Pair(o1, o2) \in accepeted$ then
        return $-1$
    end if
    if $Pair(o2, o1) \in accepeted$ then
        return 1
    end if

    $candidates \leftarrow accepted.filter\{p \rightarrow o1 \in p \text{ or } o2 \in p\}$

    for $s \leftarrow candidates.filter\{p \rightarrow p.first = o1\}$ do
        if $Pair(s.second, o2) \in candidates$ then
            $accepted.append(Pair(o1, o2))$
            return -1
        end if
    end for

    for $s \leftarrow candidates.filter\{p \rightarrow p.first = o2\}$ do
        if $Pair(s.second, o1) \in candidates$ then
            $accepted.append(Pair(o2, o1))$
            return 1
        end if
    end for

    throw $SortException(Pair(o1, o2))$
end function
```

During the sorting procedure, the next pair of elements to present to a human assessor is obtained by calling the $NEXT()$ function (Listings 2). The resulting assessment is then recorded using the $VOTE(\cdots)$ function, together with the unique *id* of the assessor.

**Algorithm 2** Mechanism to determine next pair to be compared.

```
function NEXT
    try
        for $i \leftarrow \lfloor log_2(length(elements)) - 2 \rfloor \cdots 0$ do
            $len \leftarrow \frac{length(elements)}{2^i}$
            $start \leftarrow random(length(elements) - len)$
            $sort(elements[start, start + len], compare)$
        end for
        $sort(elements, compare)$
    catch $e$
        return $e.pair$
    end try

    $e1 \leftarrow elements[random(length(elements))]$
    $e2 \leftarrow elements[random(length(elements))]$
    return $Pair(e1, e2)$
end function
```

## EXAMPLE OF IMAGES

Figure 3 shows a set of ten randomly selected images from Wikimedia Commons. Images (f), (g), and (i) are non-natural, the images are natural images (or pictures) depicting some situation, object or landscape of the real-world.

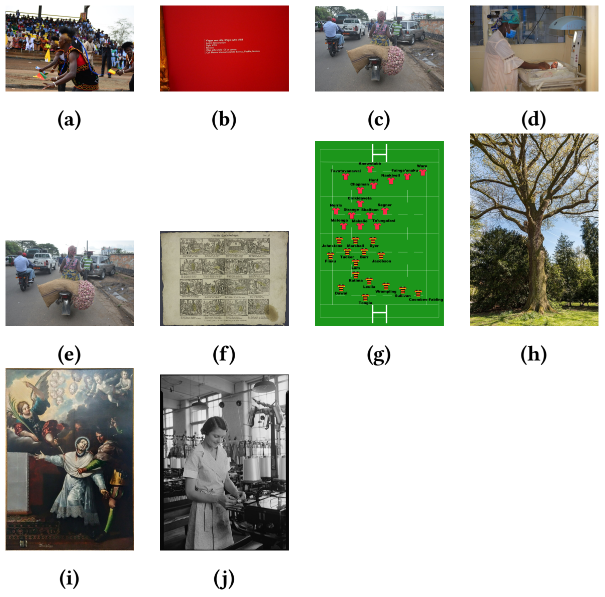

**Figure 3: Examples of natural and non-natural images in Wikimedia Commons**

## IMAGE SEQUENCES ORDERED BY DIFFERENT MEANS

To provide more context on the ordering results, we uniformly selected ten images from the crowd order (Figure 4), and identified the way these images are ordered by the other ordering mechanisms. Figure 5 shows the way these images are ranked in the 'single' order. Analogously, Figure 6 shows the images based on the 'caption' order, and Figure 8 shows them based on the 'predicted' order.

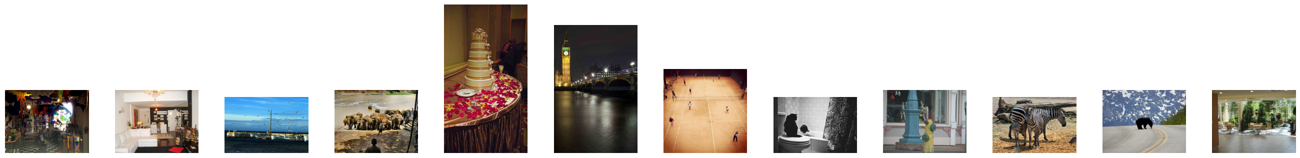

**Figure 4: Ten images uniformly selected from the 'crowd' ordering. The images are sorted from high to low VSD.**

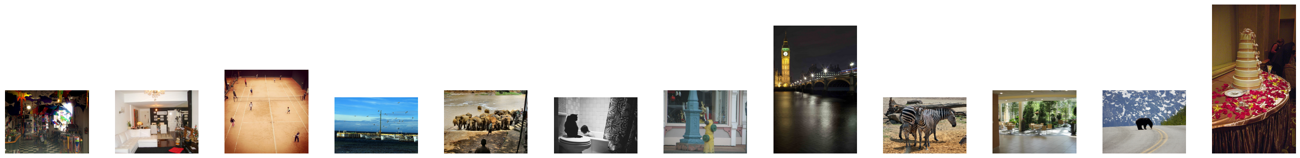

**Figure 5: Same ten images as in Figure 4 sorted as in the 'single' order. The images are sorted from high to low VSD.**

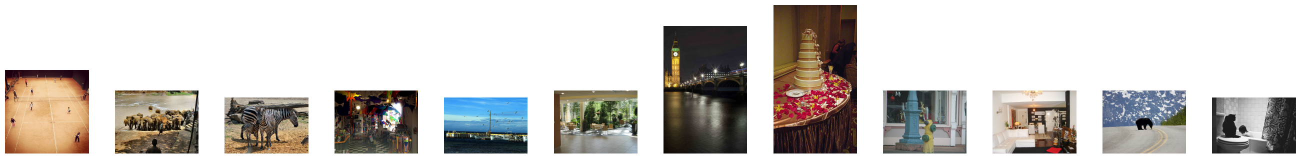

**Figure 6: Same ten images as in Figure 4 sorted as in the 'base' order. The images are sorted from high to low VSD.**

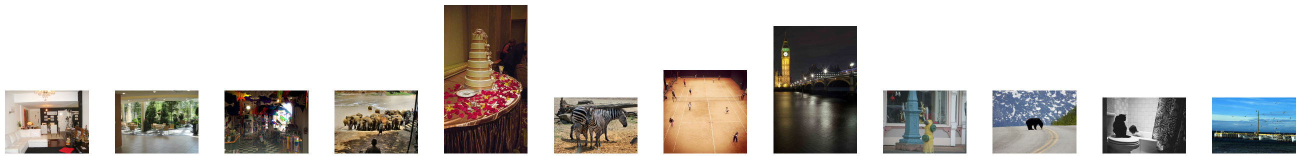

**Figure 7: Same ten images as in Figure 4 sorted as in the 'caption' order. The images are sorted from high to low VSD.**

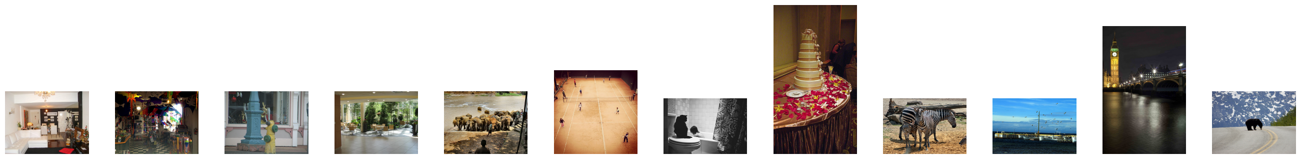

**Figure 8: Same ten images as in Figure 4 sorted as in the 'predicted' order. The images are sorted from high to low VSD.**