1. Implementation Details

In this section, we provide additional information about the implementation of each of the components of VLC-BERT.

1.1. Object Tags with YOLO

As described in Sec 3 of our paper, we utilize object tags to incorporate image context for generating commonsense inferences. In order to obtain the object tags, we use an off-the-shelf YOLO model for PyTorch, YOLOv5 by Ultralytics [2]. We use the pretrained yolov5l model to obtain object bounding boxes and the associated class name for each bounding box, on COCO 2014 and 2017 images for OK-VQA and A-OKVQA datasets respectively. We then use a confidence threshold of 0.5 to prune out objects that are unlikely to be useful. In addition, we prune out the person object name as well as the objects already present in the question phrase, to avoid unnecessary tags or repetitions. Finally, the two object names associated with the highest confidence bounding boxes are picked as the object tags for our model, O.

1.2. Knowledge Generation

As described in Sec 3.1.1 of our paper, we generate commonsense inferences from COMET [1] by inputting the question followed by “with” and two object tags into it. If S is the sentence consisting of the question and object tags and R is the relation type we want to generate from COMET, we provide it to COMET in the form S(i) R(i) [GEN] and let comet generate the commonsense inferences. Though COMET can support 50 relation types, we cherry-pick 30 relation types by removing duplicate relations and relations that are irrelevant to our work (e.g., HasPainIntensity). Table 3 (at the end of this document) provides the list of the 30 relations we used to generate commonsense expansions from COMET and the corresponding templates we used to convert COMET’s output to natural language sentences.

1.3. Knowledge Selection

As described in Sec 3.1.2 of our paper, we augment S-BERT to perform semantic search and filter and rank the relevance of commonsense inferences. In order to perform semantic search, we utilize the sentence-transformers package for SBERT [4]. We initialize our SBERT model from the pre-trained msmarco-roberta-base-ance-firstp model and train this model for 2 epochs on the training set of the corresponding task. To create the labels for this augmentation, we measure the overlap of the expansions to human annotated answers and assign a similarity score of 0.8 for overlapping expansions and a score of 0.2 for non-overlapping expansions. This augmented S-BERT model is then used to encode the question and commonsense sentences, before computing the sentence similarity between every commonsense sentence and the question, and picking the top k (K = 5) sentences to use in the input sequence of the VLC-BERT transformer.

1.4. VLC-BERT Transformer

Our implementation of the VLC-BERT transformer encoder is based on the publicly available implementation of VL-BERT [5]. The hyperparameters we use for training VLC-BERT on the VQA 2.0 (only for pre-training), OK-VQA and A-OKVQA datasets are given in Table 1.

For generating sentence embeddings for commonsense inferences that are fed into the MHA block, we use the all-mpnet-base-v2 pre-trained model from SBERT.

1.5. Implementation of Commonsense Subsets

In Sec 7.1, we describe the need for commonsense-specific subsets of OK-VQA and A-OKVQA, to show that our model improves on the baseline significantly. The lack
Table 1: Hyperparameters of our model

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>VQA P.T.</th>
<th>OK-VQA</th>
<th>A-OKVQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Size</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>Gradient Accumulation</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Epochs</td>
<td>5</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>6.25e-7</td>
<td>6.25e-7</td>
<td>6.25e-7</td>
</tr>
<tr>
<td>Visual Size</td>
<td>768</td>
<td>768</td>
<td>768</td>
</tr>
<tr>
<td>Hidden Size</td>
<td>768</td>
<td>768</td>
<td>768</td>
</tr>
<tr>
<td>Warmup Method</td>
<td>linear</td>
<td>linear</td>
<td>linear</td>
</tr>
<tr>
<td>Warmup Steps</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>MHA Heads</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>MHA Dropout</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 2: Performance of our model on OK-VQA question categories.

<table>
<thead>
<tr>
<th>Category</th>
<th>Base</th>
<th>w/ COMET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vehicles and Transportation</td>
<td>40.1</td>
<td>41.16</td>
</tr>
<tr>
<td>Plants and Animals</td>
<td>42.58</td>
<td>41.65</td>
</tr>
<tr>
<td>People and Everyday Life</td>
<td>40.09</td>
<td>39.95</td>
</tr>
<tr>
<td>Sports and Recreation</td>
<td>51.53</td>
<td>52.31</td>
</tr>
<tr>
<td>Cooking and Food</td>
<td>42.36</td>
<td>45.04</td>
</tr>
<tr>
<td>Objects, Material and Clothing</td>
<td>39.86</td>
<td>39.95</td>
</tr>
<tr>
<td>Science and Technology</td>
<td>37.38</td>
<td>38.57</td>
</tr>
<tr>
<td>Weather and Climate</td>
<td>50.7</td>
<td>48.99</td>
</tr>
<tr>
<td>Brands, Companies and Products</td>
<td>33.6</td>
<td>35.81</td>
</tr>
<tr>
<td>Geog, Hist, Language and Culture</td>
<td>40.14</td>
<td>43.4</td>
</tr>
<tr>
<td>Other</td>
<td>41.23</td>
<td>42.68</td>
</tr>
</tbody>
</table>

of any annotations for the type of reasoning required to answer the question led us to develop our own method to obtain the subsets. Below, we have the exact details required to re-create the subsets:

**Named Entities.** We use spaCy’s entity recognizer[^1]. If any word in the question or list of answers is recognized as an entity, we prune the question.

**Numerical.** We first attempt to check if a string is a number using Python’s built-in function, `isdigit()`. If it is not a digit, then we use the `word2number` package[^2] to attempt to convert words (e.g. “twenty”) into numbers. If it is successful in doing so, we deem the word to be a number. If any word in the question or list of answers is recognized as a number, we prune the question.

**Directional.** We list commonly used directional words: right, left, top, bottom, behind, under, inside, over, front, back, near, next. If any word in the question is recognized as a directional word, we prune the question.

**Symbol.** We list commonly used symbol words: logo, symbol, name, company, mascot, word, brand. If any word in the question is recognized as a symbol word, we prune the question.

**Color.** We list commonly used color words: blue, green, red, black, white, grey, purple, pink, yellow, orange. If any word in the question or list of answers is recognized as a color word, we prune the question.

**Time.** Finally, we list commonly used time words: century, year, time, month, day. If any word in the question is recognized as a time word, we prune the question.

As the task of recognizing the type of a question is challenging in itself, we tried to simplify it to a basic, reproducible method, in order to better evaluate on commonsense reasoning specific questions on the OK-VQA test set and the A-OKVQA validation set.

### 2. Evaluation

#### 2.1. Main Evaluation

The standard deviation on our scores for the OK-VQA test set is 0.20 and for the A-OKVQA validation is 0.47.

#### 2.2. Evaluation on OK-VQA Question Categories

The results provided in our paper only show the overall scores of our models on the OK-VQA [^3] dataset. In Table 2 we share the results for each question category in the OK-VQA dataset. The OK-VQA dataset has questions divided into 11 different categories[^3]. The results show that our model with external knowledge from COMET improves upon the baseline in all but three categories. Across all the models, we see that the ‘Brands, Companies and Products’ is the most challenging category, with low accuracy for both the baseline and the VLC-BERT with COMET models. This is expected, because the questions in this category often require the model to read text or symbols in the image, or identify company names and logos, which are challenging tasks outside the domain of our model.

#### 2.3. Ablations

In this section, we present additional ablations to show the impact of different components of the VLC-BERT pipeline.

[^1]: https://spacy.io/api/entityrecognizer
[^2]: https://pypi.org/project/word2number/
Ablation on number of sentences. In order to test the impact of the number of commonsense inferences $K$, we report the performance with different $K$ values. We ran our latest model with $K = 10$ and $K = 15$ sentences. On the A-OKVQA validation set, we obtain the following results: $K = 5 : 44.95$; $K = 10 : 44.57$; $K = 15 : 43.93$. We thus feed $K = 5$ commonsense inference sentences into VLC-BERT transformer, because we had observed that adding too many commonsense inferences also adds unnecessary noise in the model, which hurts performance.

Use of Object tags. In order to assess the importance of the number of object tags used in deriving commonsense inferences, we ran experiments with no (0) object tags, as well as all (>2) tags. For zero tags, we get $44.42$, and for all tags, we get $44.62$. These are slightly worse than the two tags version ($44.95$). This is in line with what we expected, since COMET is not designed to deal with complex sentences containing multiple entities, and 2 object tags stands as a good trade-off. Furthermore, in our qualitative results, we show examples of where object tags are useful in providing image context.

Impact of weak attn. supervision. Disabling weak attn. supervision, we obtain a result of $44.89$ which is slightly worse compared to $44.95$ with supervision. However, qualitative analysis shows that our model with supervision produces stronger attention weights for useful inferences compared to the model without.

3. Analysis

In this section, as mentioned in Section 7.2 of our paper we provide additional qualitative examples along with their attention scores, of where VLC-BERT improved as well as failed.

3.1. Error examples

We analyze the errors from the best version of VLC-BERT model. We randomly sample 50 erroneous examples from the validation set of A-OKVQA, analyze the errors, and classify them into five categories as shown in Figure 1. We provide an example of each category in Figure 2.

1 Visual: The model is lacking deep scene understanding that either required to answer the question, or to generate relevant commonsense inferences. This includes cases where the object tags are insufficient for describing the scene. A majority of the errors we see in VLC-BERT fall in this category, in line with our conclusions and motivations for future work on commonsense models that involve deep scene understanding.
Q: What are the riders about to do now?
Tags: motorcycle
VLC-BERT baseline: drive
VLC-BERT COMET: race
Ground Truth Answers: straighten out, tricks, flip over, skate, fall down, fall down, land, tricks, flip, tip over

Q: How are these balloons floating?
Tags: dog, couch
VLC-BERT baseline: wind
VLC-BERT COMET: magnets
Ground Truth Answers: helium, helium, helium, helium, helium, helium, helium, on air, helium

Q: What is used to pick up the suitcases?
Tags: book, suitcase
VLC-BERT baseline: cart
VLC-BERT COMET: truck
Ground Truth Answers: handle, handles, handle, handle, handle, handle, handle, handle

Q: What appliance is unhocked and placed by the sink?
Tags: oven, sink
VLC-BERT baseline: trash
VLC-BERT COMET: garbage
Ground Truth Answers: stove/oven, stove, washing machine, stove, stove, washing machine, stove, oven, stove

Q: What does it say on the boys hat?
Tags: book, bed
VLC-BERT baseline: sun
VLC-BERT COMET: happy
Ground Truth Answers: happy birthday, happy birthday, happy birthday, happy birthday, happy birthday, happy birthday, happy birthday, happy birthday, happy birthday...

Figure 2: Error analysis: We sample 50 erroneous examples from the A-OKVQA validation set, and categorize it into five categories.
Figure 3: **Qualitative examples:** (a) is from A-OKVQA, and (b) and (c) are from OK-VQA.
Table 3: Relations used for generating expansions from COMET and their corresponding sentence templates

<table>
<thead>
<tr>
<th>#</th>
<th>Relation</th>
<th>Sentence template</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AtLocation</td>
<td>You are likely to find {0} in {1}</td>
</tr>
<tr>
<td>2</td>
<td>CapableOf</td>
<td>{0} can {1}</td>
</tr>
<tr>
<td>3</td>
<td>Causes</td>
<td>Sometimes {0} causes {1}</td>
</tr>
<tr>
<td>4</td>
<td>CreatedBy</td>
<td>{1} is created by {0}</td>
</tr>
<tr>
<td>5</td>
<td>Desires</td>
<td>{0} wants {1}</td>
</tr>
<tr>
<td>6</td>
<td>HasA</td>
<td>{0} has {1}</td>
</tr>
<tr>
<td>7</td>
<td>HasFirstSubevent</td>
<td>The first thing you do when you {0} is {1}</td>
</tr>
<tr>
<td>8</td>
<td>HasProperty</td>
<td>{0} is {1}</td>
</tr>
<tr>
<td>9</td>
<td>HinderedBy</td>
<td>{0} is hindered by {1}</td>
</tr>
<tr>
<td>10</td>
<td>IsA</td>
<td>{0} is {1}</td>
</tr>
<tr>
<td>11</td>
<td>isAfter</td>
<td>{0} happens before {1}</td>
</tr>
<tr>
<td>12</td>
<td>isBefore</td>
<td>{1} happens before {0}</td>
</tr>
<tr>
<td>13</td>
<td>LocatedNear</td>
<td>{0} is located near {1}</td>
</tr>
<tr>
<td>14</td>
<td>MadeOf</td>
<td>{0} is made of {1}</td>
</tr>
<tr>
<td>15</td>
<td>MadeUpOf</td>
<td>{0} is made up of {1}</td>
</tr>
<tr>
<td>16</td>
<td>NotCapableOf</td>
<td>{0} is not capable of {1}</td>
</tr>
<tr>
<td>17</td>
<td>NotHasProperty</td>
<td>{0} does not have the property of {1}</td>
</tr>
<tr>
<td>18</td>
<td>NotIsA</td>
<td>{0} is not {1}</td>
</tr>
<tr>
<td>19</td>
<td>NotMadeOf</td>
<td>{0} is not made of {1}</td>
</tr>
<tr>
<td>20</td>
<td>ObjectUse</td>
<td>{0} is used for {1}</td>
</tr>
<tr>
<td>21</td>
<td>PartOf</td>
<td>{1} has {0}</td>
</tr>
<tr>
<td>22</td>
<td>SymbolOf</td>
<td>{0} is a symbol of {1}</td>
</tr>
<tr>
<td>23</td>
<td>UsedFor</td>
<td>{0} is used for {1}</td>
</tr>
<tr>
<td>24</td>
<td>xAttr</td>
<td>{0} is seen as {1}</td>
</tr>
<tr>
<td>25</td>
<td>xEffect</td>
<td>{0} then {1}</td>
</tr>
<tr>
<td>26</td>
<td>xIntent</td>
<td>Because {0} wanted {1}</td>
</tr>
<tr>
<td>27</td>
<td>xNeed</td>
<td>Before {0} needed {1}</td>
</tr>
<tr>
<td>28</td>
<td>xReact</td>
<td>As a result {0} feels {1}</td>
</tr>
<tr>
<td>29</td>
<td>xReason</td>
<td>{0} reasons {1}</td>
</tr>
<tr>
<td>30</td>
<td>xWant</td>
<td>As a result {0} wants {1}</td>
</tr>
</tbody>
</table>
References


