BLIP-2 Classifier: Using generative models on classification tasks

Razvan Florian Vasile
University of Bologna
Department of Computer Science
razvanflorian.vasile@studio.unibo.it

Abstract

A lot of attention has been given in recent years to the development of complex architectures designed to integrate multimodal capabilities. While extending existing infrastructures with more and more intricate and complex modules, some of the more common use-cases may not be immediately applicable. While popular libraries like Transformers offer rich APIs, they often lack direct support for common classification tasks when using multimodal models like BLIP-2. To address this limitation, I adapt the BLIP-2 architecture, which was originally designed for question-answering, to perform classification tasks within the Transformers library.

The algorithm is evaluated using two datasets: Easy-VQA and Daquar. Easy-VQA contains simple questions about geometric shapes, while Daquar is more challenging, requiring answers to questions about objects in indoor scenes. The model achieves 91% accuracy on Easy-VQA and 78% accuracy on Daquar. UMAP visualization of the learned features suggests that the model is learning effectively, particularly for Easy-VQA where the model shows higher accuracy. The project is available at: https://github.com/atomwalk12/VisualQA.

1. Introduction

Problem description. BLIP-2 [6] is a generative model that uses an architecture consisting of a Visual Encoder, a Q-Former and an LLM. The model's application on classification tasks is not immediate and as a result may be daunting to train for unexperienced users.

For the sake of curiosity, a user may find it interesting to understand the effectiveness of generative models on datasets tailored for classification tasks. Beyond the practical benefits of not being limited to a specific dataset type, a classification problem may lead to insight of how the model functions and understand the practical limitations of the architecture.

Task significance. Effectively utilizing generative models like BLIP-2 for classification opens up new possibilities for utilizing pre-trained models on a wider range of tasks, potentially reducing the need for extensive task-specific training data, leveraging the knowledge acquired during the model's pre-training phase.

Moreover, BLIP-2 is a powerful model [2] that does not require significant computational resources to fine-tune. This is due to the fact that the architecture contains a Visual Encoder and an LLM with trainable weights that are kept frozen during training. The model's fine-tunability on modest hardware makes it accessible to a wider range of users, increasing its potential applications and encouraging further experimentation.

Limitations of existing approaches. The current HuggingFace implementations do not provide end-to-end classes that are directly applicable to classification tasks [4]. This makes training classifiers on these models not immediate and may require knowledge of the intricate workings of the model in order to extend to other learning objectives. Although new methods like `Blip2VisionModelWithProjection` and `Blip2TextModelWithProjection` have been introduced, they are not immediately usable and require additional modifications.

Key ideas and results. Our approach involves extracting visual features from the Vision Encoder, combining them with textual data from the Q-Former, and training a classification network on these combined representations. Experiments on the Easy-VQA and DAQUAR datasets show that the proposed learning algorithm adapted from the BLIP-2 model achieved good classification accuracy, overcoming challenges encountered during the training of the generative baselines, such as overfitting early during the first training epoch.

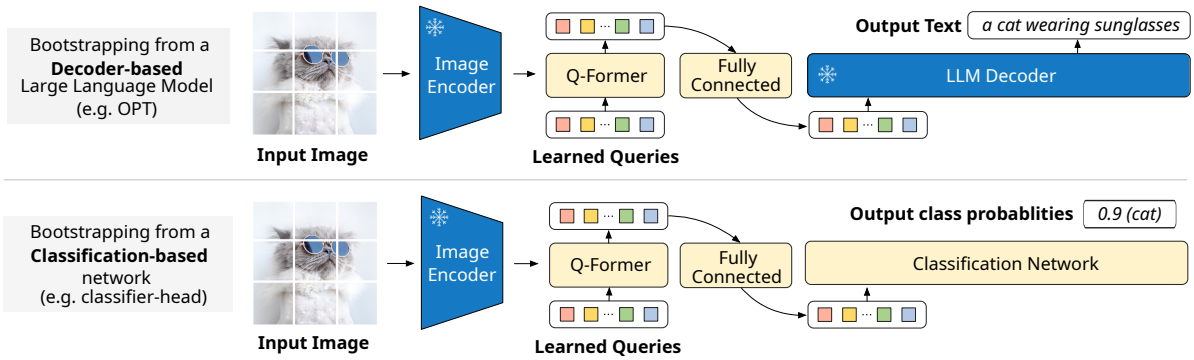


Figure 1. Figure adapted from the original paper [6]. A link to the license is available here. The upper image represents the architecture of the baseline model involving a frozen vision encoder and a frozen LLM decoder. The lower image displays our proposed architecture.

2. Related work

2.1. Baselines

Reference paper. The inspiration for this project is due to the the paper «BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models » [6], which presents a generative model used for visual question-answering and image captioning tasks.

The model architecture is composed of a frozen Vision Encoder and a frozen LLM. During the fine-tuning process, a special module named Q-Former is used to bridge the gap between the semantic information present in images with the knowledge reflected in text. The architecture is illustrated in Figure 1. Please note that only the Q-Former has learnable weights and as a result the computational requirements necessary to train the network are greatly diminished, making the training process tractable on consumer-grade machines.

Wider research context. Below I discuss similar models presented in the literature highlighting their strengths and weaknesses.

CLIP. The CLIP [13] architecture employs contrastive learning to align image and text representations. This approach enables the model to learn to position similar data points in close proximity and dissimilar ones at a distance. BLIP-2 leverages contrastive learning as well, however the key advantage relates to using pre-trained frozen image encoders for efficient feature extraction. By decoupling the dependency between the LLM, the vision encoder and the Q-Former this leads to compute-efficient training where only a submodule needs to be fine-tuned. This is not the case for CLIP as it requires end-to-end training.

Flamingo. On the other hand, Flamingo's [1] architecture involves inserting new cross-attention layers into a large language model, allowing the LLM to incorporate visual information. By training the LLM directly, this leads to adapting the network to a wide range of tasks at the expanse of more expansive training computational requirements. BLIP-2 emphasizes the use of pre-trained knowledge and the adjustment of a lesser number of parameters during fine-tuning.

Why this paper? BLIP-2 has an unique architecture that is designed to leverage an efficient use of trainable parameters making it superior to the alternative models. This leads to efficient fine-tuning with reduced computational requirements. Also, this allows users to reuse all the knowledge acquired during pre-training while having to fine-tune only a portion of the weights.

As an example, BLIP-2 outperforms Flamingo 80B on zero-shot VQAv2 dataset with 54 times fewer trainable parameters [6]. The size allows faster experimentation and exploration of larger datasets. Additionally, BLIP-2 outperforms other similar models on various benchmarks [2].

2.2. Transformers library

Limitations of current work. Currently, the HuggingFace documentation [4] supports three classes that may be useful for tackling classification problems. However, none is a perfect match.

Blip2ForImageTextRetrieval. This architecture is primarily designed for image-text retrieval tasks, where the goal is to match images with corresponding text descriptions. It's not structured to output multiple labels or class probabilities for a given input.

Blip2TextModelWithProjection. This model is focused on processing text inputs and projecting them into a shared embedding space. It doesn't have components specifically tailored for classifying images into multiple categories or assigning multiple labels.

Blip2VisionModelWithProjection. Similar to the text model, this architecture processes visual inputs and projects them into an embedding space. While it can extract features

Hyperparameter	Value
LoRA Rank	Generator: 1, Classifier: 8
Optimizer	AdamW
Learning Rate	1e-4
Weight Decay (L2)	1e-6
L1 Regularization Strength (λ_{L1})	0.01
Learning Rate Scheduler	CosineAnnealingLR
Scheduler T_max	500
Scheduler min_lr	1e-6

Table 1. Fine-tuning hyperparameters.

from images, it lacks the necessary output layers for classification tasks. Also, the resulting features are not immediately compatible due to their different dimensional spaces.

Conclusion. The BLIP-2 model, as described in the documentation, is primarily designed for tasks like image captioning, visual question answering, and image-text retrieval. These tasks involve generating text based on image inputs or matching images and text, rather than classifying images into predefined categories. As a result, the existing methods lack support for classification tasks.

3. Approach

3.1. Baselines

Efficient Training. `PEFT` (Parameter-Efficient Fine-Tuning) [18] was used to efficiently fine-tune the model. It involves various techniques aimed at fine-tuning large pre-trained models while minimizing computational costs and memory requirements.

To be more specific, I utilized a specific PEFT technique called `LoRA` (Low-Rank Adaptation) [3]. This PEFT method introduces wrappers around some of the weights, the so called adapters, that allow to be fine-tune the model for task-specific adaptation while keeping the primary model parameters frozen, significantly reducing the number of trainable parameters.

Hyperparameters. During the experiments, one recurring issue involved the generative's model early overfitting on training data during the first training epoch. To alleviate this issue, special hyperparameters were adopted, which are reported in Table 1 and described below:

- **LORA Rank.** The rank is reduced to 1 in order to use fewer learnable parameters, implying less expressive power and more constrained updates and as a result a lesser chance of overfitting. This maximizes the model's reliance on pre-trained data.
- **L2 regularization and the AdamW optimizer.** The implicit `L2` regularization encourages all weights to become smaller by applying a penalty, with a stronger effect on larger weights than on smaller weights.
- **L1 regularization [16]** By adding a penalty term to the loss based on the absolute value of the weights, L1 regularization encourages sparsity in the model. This can drive some weights to exactly zero. As a result, less important features may be completely eliminated.
- **Learning rate scheduler.** The `CosineAnnealingLR` balances larger steps and smaller steps through periodic transitions between high and low learning rates. This cyclic behaviour helps the exploration of the loss landscape, making it more difficult to get stuck in a local minima.

BLIP-2 Architecture. The frozen and trainable components are shown in Figure 1. In the original architecture, the Q-Former is the trainable module that bridges the gap between the frozen image encoder and the frozen language model. Our approach is to introduce a trainable classification network on-top of the Q-Former. This adds flexibility as it allows to easily determine the complexity of the trainable network based on the complexity of the problem.

The Q-Former Architecture. The Q-Former utilizes 32 learnable query embeddings, each with dimension 768 [6], used to extract visual features from a frozen image encoder. This results in a total output query representation of size 32 x 768, significantly smaller than the typical size of the frozen image features (e.g., 257 x 1024). This way the queries act as an information bottleneck yielding the emergent property that the network will focus on the relevant visual information and reduce the burden on the LLM, in the case of the generative model, and the classifier, when using a classification network.

3.2. Algorithm

Prompt format. A special prompt format is required to train the model[1]. For the generative task, the model sees the question and the associated answer. Conversely, in the case of the classifier, a one-hot vector is used.

The approach. The approach combines the vision and textual features from the Vision Encoder and Q-Former respectively. Then the features are projected into a common multi-dimensional space, and a feed-forward network is used to perform the classification.

The classier pipeline is shown in Algorithm 1 and described below.

1. It starts by extracting features from the input image using a vision model.
2. These image features are then improved using a Q-Former, which adds context to the image features based on learned query tokens.

[1]Prompt format: «Question: {}? Answer: {}»

Algorithm 1 Feature Extraction Algorithm

```
def forward(pixel_values, input_ids):
  # 1. Extract image features
  img_embeds = vision_model(pixel_values)

  # 2. Use the Q-Former. This step adds context to the
  # image features based on learned query tokens
  image_embeds = qformer(query_tokens, img_embeds)

  # 3. Extract text features.
  text_embeds = text_embeddings(input_ids)

  # Add context using learnable Q-Former query tokens
  question_embeds = qformer(text_embeds)

  # 4. Project the features to a lower dimension.
  # Position 0 is the representation of the sequence
  text_proj = text_projection(question_embeds[:,0,:])
  vision_proj = vision_projection(image_embeds)

  # 5. Normalize the features to facilitate convergence
  text_repr = normalize(text_proj)
  image_repr = normalize(vision_proj)

  # 6. Flatten and combine features into a single vector
  image_repr_flat = flatten(image_repr)
  combined = concatenate(text_repr, image_repr_flat)

  # 7. Classification
  logits = classifier(combined)

  # 8. The results be used with a sensible loss function
  return logits, text_repr, image_repr
```

3. For the text input, it first generates embeddings from the input IDs, then processes these through the Q-Former to add context related to the vision features.
4. Both text and image embeddings are then projected to a lower dimension. For text, it only uses the first token's embedding, which is typically a special classification token representative of the entire sequence.
5. The normalization is done to facilitate convergence by ensuring the internal covariate shift problem is mitigated.
6. The image representation is flattened, and then concatenated with the text representation to create a combined feature vector.
7. Finally, this combined feature vector is passed through a classifier to produce logits (raw prediction scores for each class).
8. The function returns these logits along with the processed text and image representations.

This process allows the model to make predictions based on both textual and visual information, combining them in a way that can capture relationships between the two modalities. The architecture of the classifier is customizable and a representation of the proposed classifier architecture is shown in Figure 2.

Dataset Statistics				
	Easy-VQA		DAQUAR	
Metric	Original	Processed	Original	Processed
Total number of items	48248	16905	12468	13722
Q1	813.0	815.0	2.0	654.5
Q2	826.0	833.0	4.0	757.5
Q3	2105.0	2091.0	13.75	839.25
IQR	1292.0	1276.0	11.75	184.75
Lower Whisker	0	0	-15.625	377.375
Upper Whisker	4043.0	4005.0	31.375	1116.375
Number of unique labels	13	13	582	18
Mean frequency	3711.38	1300.38	24.62	762.33
Median frequency	826.00	833.00	4.00	757.50
Number of outliers	2	0	92	0

Table 2. Statistics on the Easy-VQA and DAQUAR datasets.

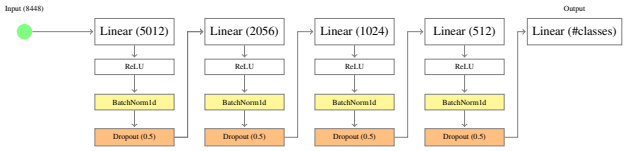


Figure 2. Classifier neural network architecture.

4. Experiments

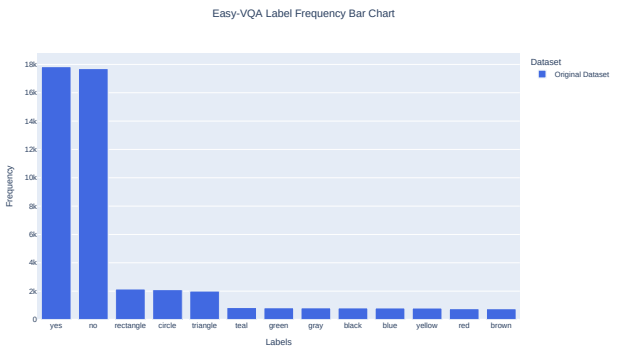
4.1. Easy-VQA dataset

Dataset composition. The dataset consists of answering queries about geometrical shapes present in images. These can include answers related to the color, shape or binary yes/no responses, as illustrated in Figure 9.

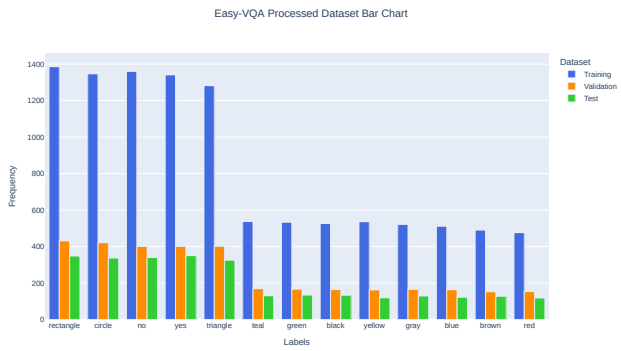
Interpretation of the figures. We visualize the dataset in order to identify potential problems that might compromise the model's ability to generalize. For this, bar charts are used to check the distribution of the classes and box-plots are used to identify outliers. The two plots are shown in Figures 3a and 3c.

From the bar chart, it quickly becomes clear that the dataset is skewed, where two classes are over-represented. This intuition is then enforced by the statistical data recovered from the box-plot, where we can identify outliers that would compromise the generalization capabilities of the model unless regularization methods are used. To form a more precise idea about how significant these problems are, statistical information is mentioned in Table 2.

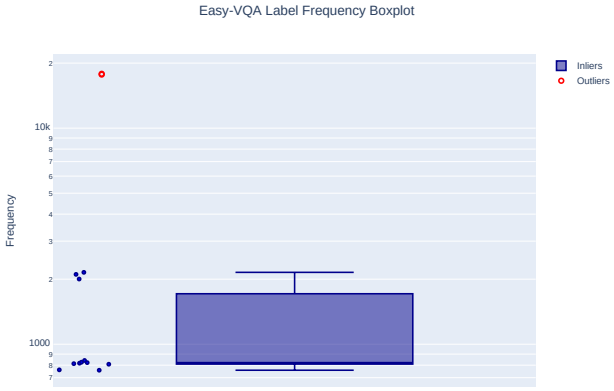
Statistical analysis. From Table 2, we can see that Quartile 3 is defined at around 2.1k samples, while the two outliers – the yes and no classes – have close to 18k entries. Moreover, most label frequencies are clustered near



(a) Bar-chart prior to processing (interactive version).



(b) Bar-chart postprocessing (interactive version).



(c) Box-chart prior to processing (interactive version).

(d) Box-chart postprocessing (interactive version).

Figure 3. Illustrations of the EasyVQA dataset before and after processing.

the lower end of the distribution. These observations suggest that the two overrepresented classes will likely cause generalization biases where the model will be able to predict very well the outliers, while the remaining classes at the lower end of the distribution will not be predicted consistently.

Preprocessing. The preprocessing phase involves removing samples that cause these outliers to be disproportionately represented. This ensures a more balanced model, able to generalize well across all possible question-answering classes.

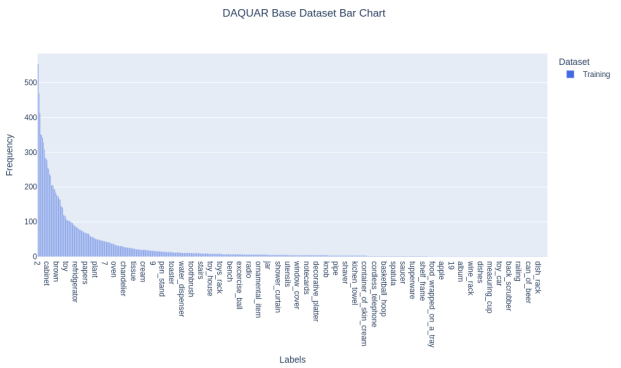
This process significantly reduced the total number of samples from 48,248 to 16,905, primarily by downsampling the overrepresented 'yes' and 'no' categories. This eliminated the outliers and balanced the dataset, evening out the other metrics shown in Table 2. The box plot and bar chart from Figures 3b and 3d visually confirm these changes, showing that the resulting changes are local to the two outlier categories. Effectively the samples have been distributed evenly across all categories. This would likely lead to better model performance and generalization accuracy.

4.2. DAQUAR Dataset

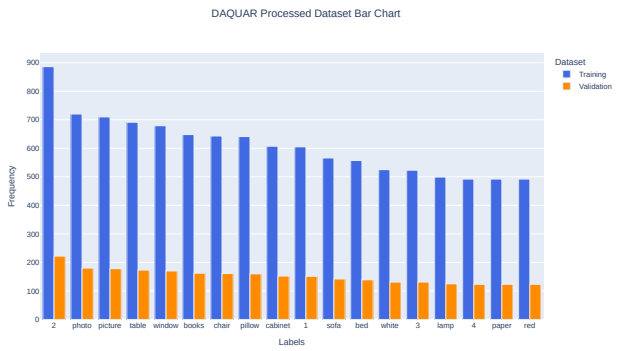
Dataset composition. The DAQUAR dataset is more challenging than the Easy-VQA dataset, the human baseline accuracy being reported to be 60% [8]. The questions are related to properties of indoor objects such as furniture type, color or object count. Sample question-image pairs are shown in Figure 11.

Interpretation of the figures. Bar charts are used to analyze the class distribution, while box-plots help identify outliers. These visualizations are presented in Figures 4a and 4c.

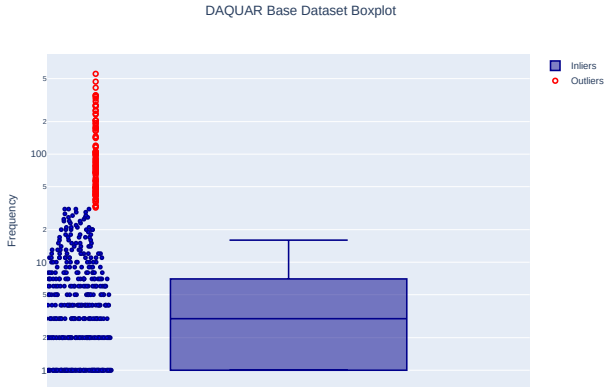
The bar chart suggests a long-tailed distribution and significant class imbalance, with a median of 4 examples per class. This observation is supported by the box-plot analysis, which identifies 92 outliers representing classes with a high sample count. This skewed distribution, where the majority of classes have very few examples while a small



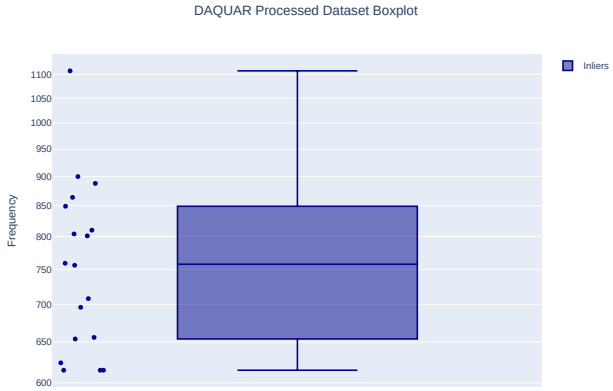
(a) Bar-chart prior to processing (interactive version).



(b) Bar-chart postprocessing (interactive version).



(c) Box-chart prior to processing (interactive version).



(d) Box-chart postprocessing (interactive version).

Figure 4. Illustrations of the DAQUAR dataset before and after processing.

subset contains significantly more, suggests potential challenges for model training and generalization across classes.

Statistical analysis. Table 2 further emphasizes how challenging the dataset is, particularly around the under-represented and over-represented classes. Moreover, the table suggests that there exist 92 outliers and that the median frequency of the number of examples per class is 4, while the mean frequency is 24. This suggests that a thorough preprocessing phase is required to even out the sample distribution in order to achieve good generalization across all classes.

Preprocessing. Given the sparsity and training challenges the dataset, some assumptions were made that allow to effectively assess the feature extraction algorithm. Initially, the dataset consisted of 12468 samples which were preprocessed the following way:

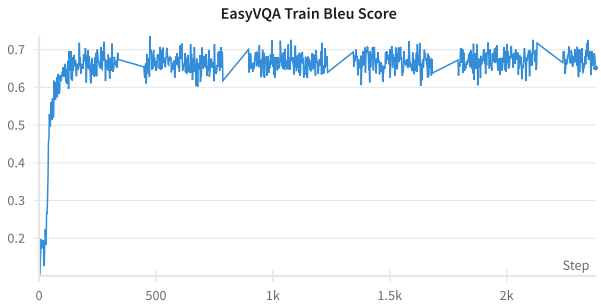
- Reducing the number of classes from 582 to 18. This gives the possibility of visualizing the results and to evaluate the semantics of the generated classifier logits.
- The total of samples were increased from 5359 to 13722 through data augmentation.
- A 80-20 stratified split to ensure that the training and validation sets are representative of the original dataset.

The bar chart and box plot from Figures 4b and 4d show the post-processed dataset. They illustrate an even distribution across all classes, with the median frequency increasing from 4 to 757.5 samples per class as suggested by Table 2. This balanced distribution allows a more direct comparison of the results with those obtained for the Easy-VQA dataset.

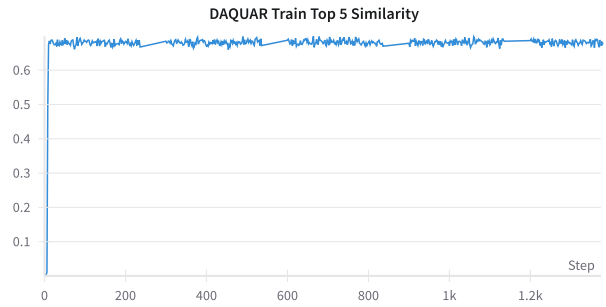
5. Analysis

5.1. Baselines

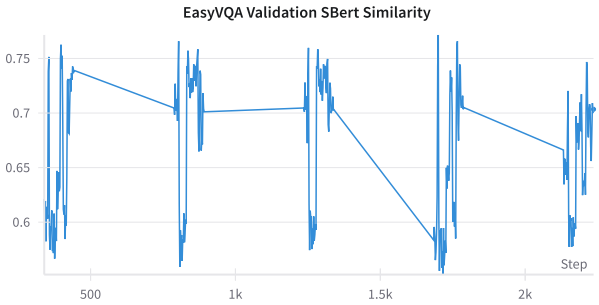
Baselines. The results for the baseline models are shown in Figure 5. The classification tasks performed generally better than the generative baselines. The issues encountered while training the baselines involved early overfitting on the training data during the first epoch. This problem was circumvented using a classifier of customizable com-



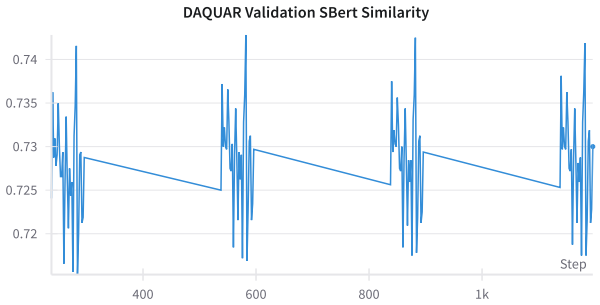
(a) Accuracy for the EasyVQA training split.



(b) Accuracy for the DAQUAR training split.



(c) SBert similarity for the EasyVQA validation split.



(d) SBert similarity for the DAQUAR validation split.

Figure 5. Accuracy during training/validation on the baseline generator models. Additional plots are available here.

plexity. The accuracy reached on the validation split for the EasyVQA and DAQUAR datasets are 91% and 78% respectively. This contrasts with the results reached for the EasyVQA and DAQUAR generative baselines reaching 70% and 73% respectively.

5.2. Error analysis

This section makes brief comments on a few selected evaluation metrics quantifying the effectiveness of the learning method described in Algorithm 1.

EasyVQA. Figures 6a and 6c suggest that the model learns effectively early in training and maintains stable performance. It is also visible how the loss function steadily decreases, while the F1 score on the validation set is relatively high after the first epoch, reaching a score close to 70%-80%. This score subsequently improves to 90% later during the training phase. This indicates that the dataset is not as difficult to process after the preprocessing phase which adequately smoothed out the class distribution leading to effective training and good generalization accuracy.

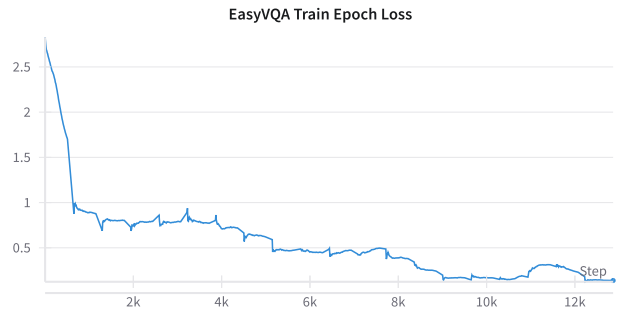
Moreover, regarding the EasyVQA dataset the confusion matrix on the evaluation split is displayed in Figure 7a. The figure suggests that the model learnt good representations of the classes in the dataset, effectively reaching good results across most of the categories, with errors around the yes and no binary answers.

Challening categories. Figure 10 displays individual examples of the yes/no classes. The model had difficulties classifying these two classes likely due to the semantic similarities between these two words and how the questions were phrased.

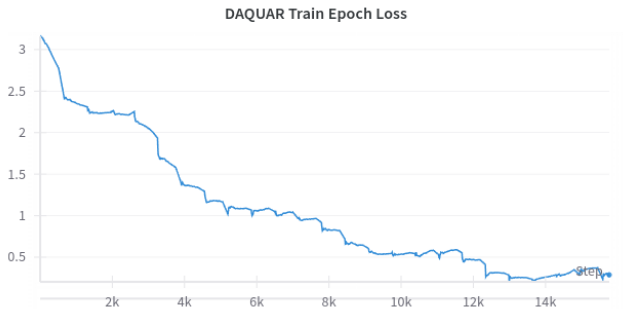
DAQUAR. The training outcomes are shown in Figures 6b and 6d. The diagram suggests that the model loss gradually improves from 3 to 0.3. The validation F1 score starts low (20%-30%) and improves consistently, eventually settling around 70%-80%. These metrics indicate DAQUAR to be a more challenging dataset than EasyVQA, requiring longer training time and achieving lower overall performance.

The confusion matrix is shown in Figure 7b. The onset impression is that the model categorizes examples fairly well. For instance, the performance on household categories such as the bed, chair or sofa is relatively high (see examples in Figure 11). In contrast, the classes that pose difficulties are the numerical categories, as illustrated in Figure 12, suggesting that the model cannot count quantities well.

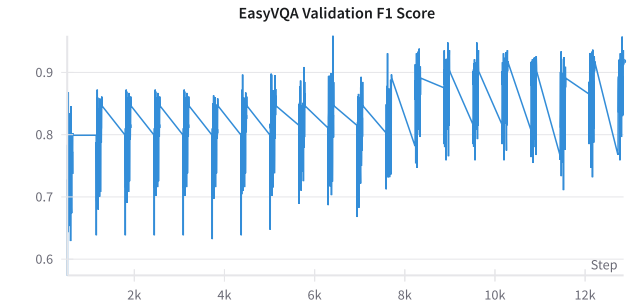
Challenging categories. The images show scenes with objects which appear difficult to be easily identified even for human observers. In fact the human accuracy baseline report for this dataset is approximately 60% [8]. Physical shapes, like 'bed', 'chair' or 'sofa', are more easily iden-



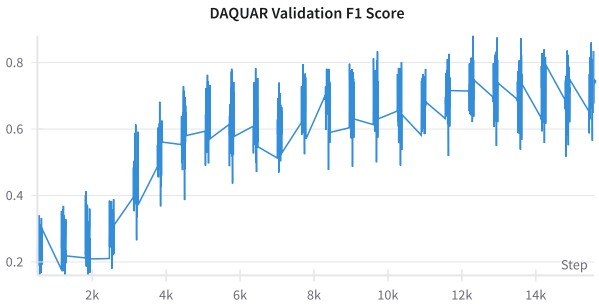
(a) Loss for the EasyVQA training split.



(b) Loss for the DAQUAR training split.



(c) F1 score for the EasyVQA validation split.



(d) F1 score for the DAQUAR validation split.

Figure 6. Accuracy during training/validation on the classifier models. Additional plots are available here.

tified, while numerical quantities are not. This can be attributed to the vision encoder's frozen pre-trained weights, which excels at individual object recognition but lacks specific training for quantitative relationships between multiple objects in a scene.

5.3. Ablation studies

Classification models. Multiple ablation experiments were carried out to analyze the effectiveness of each individual component involved in the classification process. Below I outline some of the key experiments:

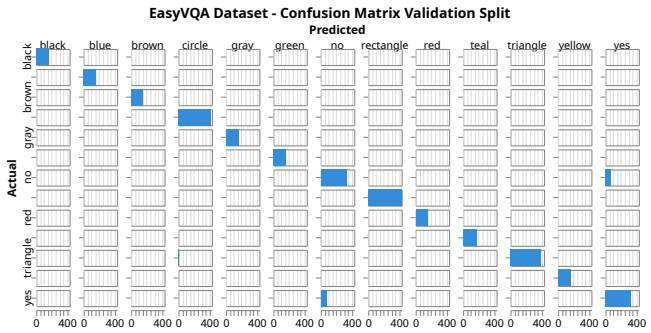
- The first experiment (link) involved using an MLP classifier on-top of the base model. Used the last hidden state of the `Q-Former` and the last hidden state of the vision encoder. Then the mean across the first dimension of the q-former is used to make the visual and textual features' dimensions compatible.
- Second experiment (link) involved projecting the two intermediate dimensions into a common space by using linear layers, rendering the features homogeneous. This contrasts with the Experiment 1 which used a mean average to make the features compatible.
- Third experiment (link) uses the `pooler_output` of the Q-Former instead of the last hidden state. The pooler_output is provided through the `Blip2ForConditionalGeneration` as shown in the Transformers library [4].
- Last experiment (link) used the `Blip2Model` base class instead of `Blip2ForConditionalGeneration`. This uses a base class which doesn't rely on the generative model for fetching the features.

None of these algorithms performed better than the final solution [17] proposed in Algorithm 1.

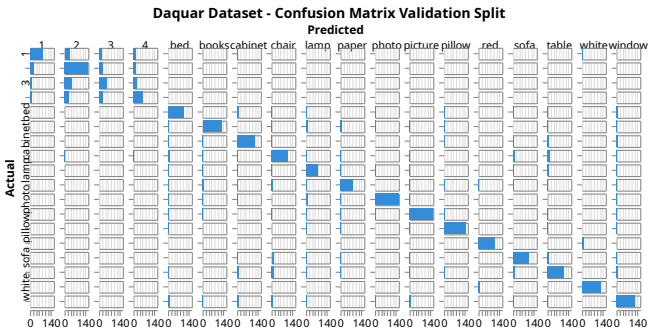
5.4. Embeddings visualization.

During the training process, the classifier embeddings were stored for subsequent visualization. This can help build an intuition on how training progresses and check whether the semantics across the learnt embeddings are well identified by the model. The resulting embeddings are projected into a 2-dimensional space using UMAP [9] and the results for the EasyVQA and Daquar datasets are shown in Figures 8a and 8b respectively.

EasyVQA dataset. Figure 8a depicts the embeddings generated during the 15^{th} epoch during the training process. The resulting embeddings for the EasyVQA dataset have categories with clear cluster boundaries, where only a few groups contain mixed-in elements. This is encouraging feedback as it shows that the model was able to create internal representations of these embeddings in a way that clearly identifies the concepts involved, which shows that



(a) EasyVQA dataset.



(b) DAQUAR dataset.

Figure 7. Confusion matrices on the validation sets.

the feature extraction algorithm is effective. However, some clusters are still not entirely homogeneous, which suggests that misclassifications are still possible.

For example, the clusters identifying colors ('blue', 'teal', 'gray', 'brown' and 'black') are projected nearby to each other while the ones identifying shapes are projected further away ('rectangle', 'circle' or 'triangle'). Furthermore, binary choices like 'yes'/'no' are identified nearby to each other, however further away from the other categories.

Daquar dataset. Similar characteristics can be identified for the DAQUAR dataset shown in Figure 8b, where the 13^{th} epoch is shown. However, in this case the emerging clusters have fuzzier borders, where multiple categories are present in distinct clusters. Particularly the left region of the plot, suggests the model struggles to form clear decision boundaries for certain categories, which aligns with the dataset's known complexity and lower human performance baseline.

Nevertheless, there are encouraging aspects as well. For instance, numbers tend to be placed within the same shared cluster. This suggests that while the model might have difficulties identifying numerical values, it clearly identifies the concept of a digit.

6. Ethical considerations

Ethical challenges. Below are highlighted some ethical challenges around the misuse of the library or potential ethical implications and risks.

Electricity consumption. Firstly, one possible concern is about the energetic power consumption of these models. It is known that these systems consume a huge amount of electricity, especially during the pre-training phase but during fine-tuning and inference as well. Practical policies need to be set up to minimize the impact these algorithms have on the environment.

Pre-trained dataset biases. The reliance on frozen pre-trained components (vision encoder and LLM) with limited fine-tuning through PEFT and LoRA introduces bias concerns. While UMAP visualizations show effective clustering of similar concepts, the model struggles with abstract relationships, particularly numerical quantities. This is evident in the DAQUAR dataset, where accuracy on numerical questions (70-80%) suggests inherited biases from the frozen components. Users should note that while frozen pre-trained components offer efficiency gains, they may inherit and amplify existing model biases.

Mitigation techniques. Now, I would like to highlight potential solutions to the identified problems.

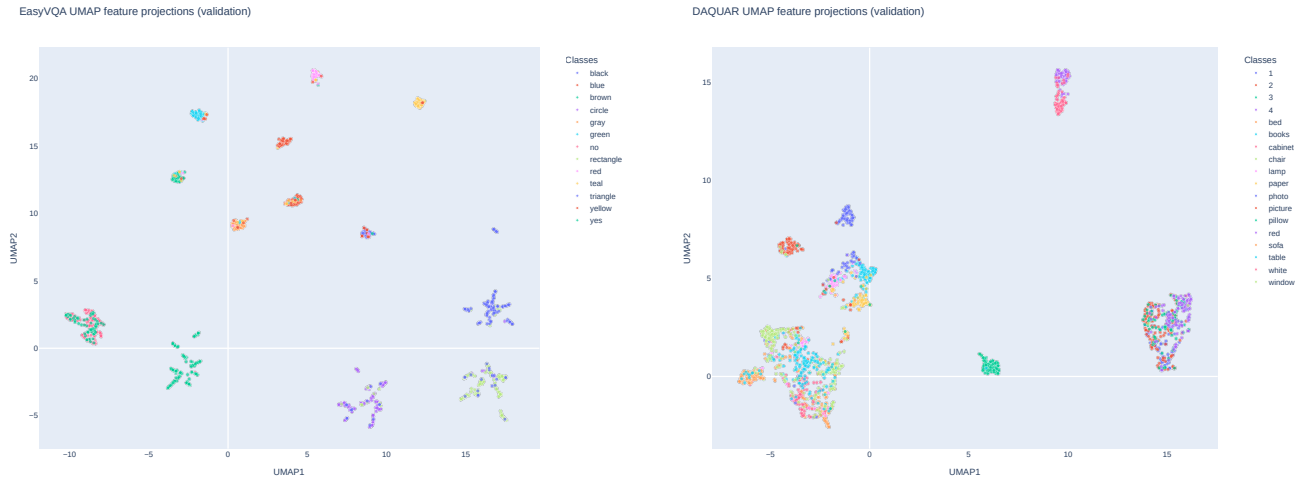
Electricity consumption. Firstly, addressing the initial concern, [14] and [7] suggest practical methods for reducing energy consumption during inference. Solutions such as power capping and model compression optimizations are highlighted. These advancements, are unlikely to significantly affect the overall model performance [5]. Consequently, using smaller models should not negatively impact the overall user experience.

User education. Moreover, addressing the bias concerns requires developing educational resources that clearly communicate the model's limitations. Users should be informed about the inherited biases from frozen pre-trained components, particularly regarding abstract relationships and numerical quantities. This transparency can help prevent over-reliance on the model's outputs and enable users to better understand when the model's predictions may be less reliable.

7. Conclusion

Future work. Below are some potential ideas for exploration and improvement:

- Assess additional modalities to include text [12], graph [11] or audio classification [10]. These use-cases change the semantics of the extracted features leading to new interpretations and potential insights into their strengths.
- Investigate the impact of freezing versus fine-tuning the vision model and the LLM on downstream tasks. This



(a) EasyVQA dataset (interactive version and video).

(b) DAQUAR dataset (interactive version and video).

Figure 8. The distribution of the embeddings extracted from the validation datasets.

could lead to insights into the effectiveness of the embeddings generated by the Q-Former and their mutual dependency on the learning process.

Limitations. The effectiveness of the model on more challenging problems has to be checked. Although, there have been reached good results with respect to the expressiveness of the feature extraction pipeline the DAQUAR dataset was simplified. Both the EasyVQA and DAQUAR datasets reached better accuracy with respect to the baselines, however further analysis should be made to identify the extent to which the algorithm is expressive enough to effectively solve more challenging classification problems.

Key findings. To conclude, this paper demonstrates a training method for training generative models for classification tasks using the Blip-2 architecture. By leveraging learnable Q-Former embeddings, a frozen visual encoder and a classifier, it was possible to train a relatively large model consisting of over 2.7B parameters on off-the-shelf computers by leveraging efficient training methods such as PEFT and LoRa.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning, 2022. 2

[2] BLIP-2. Papers with Code - BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models — paperswithcode.com. https://paperswithcode.com/paper/blip-2-bootstrapping-language-image-pre, 2023. [Accessed 13-08-2024]. 1, 2

[3] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021. 3

[4] Huggingface. BLIP-2 — huggingface.co. https://huggingface.co/docs/transformers/model_doc/blip-2, 2024. 1, 2, 8

[5] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference, 2017. 9

[6] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023. 1, 2, 3

[7] Sasha Luccioni, Yacine Jernite, and Emma Strubell. Power hungry processing: Watts driving the cost of ai deployment? In *The 2024 ACM Conference on Fairness, Accountability, and Transparency*. ACM, 2024. 9

[8] Mateusz Malinowski and Mario Fritz. A multi-world approach to question answering about real-world scenes based on uncertain input, 2015. 5, 7

[9] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018. 8

[10] PapersWithCode. Papers with Code - Audio Classification — paperswithcode.com. https://paperswithcode.com/task/audio-classification, 2024. [Accessed 26-10-2024]. 9

[11] PapersWithCode. Papers with Code - Graph Classification — paperswithcode.com. https://paperswithcode.

com/task/graph-classification, 2024. [Accessed 26-10-2024]. 9

[12] PapersWithCode. Papers with Code - Text Classification — paperswithcode.com. https://paperswithcode.com/task/text-classification, 2024. [Accessed 26-10-2024]. 9

[13] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 2

[14] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. From words to watts: Benchmarking the energy costs of large language model inference, 2023. 9

[15] Ranu Sewada, Ashwani Jangid, Piyush Kumar, and Neha Mishra. Explainable artificial intelligence (xai). *international journal of food and nutritional sciences*, 2023.

[16] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996. 3

[17] Razvan Vasile. VisualQA/lib/models/blip2_classifier_experiment5.py at main ů atomwalk12/VisualQA — github.com. https://github.com/atomwalk12/VisualQA/blob/main/lib/models/blip2_classifier_experiment5.py, 2024. [Accessed 01-11-2024]. 8

[18] Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment, 2023. 3

8. Appendix: Dataset images

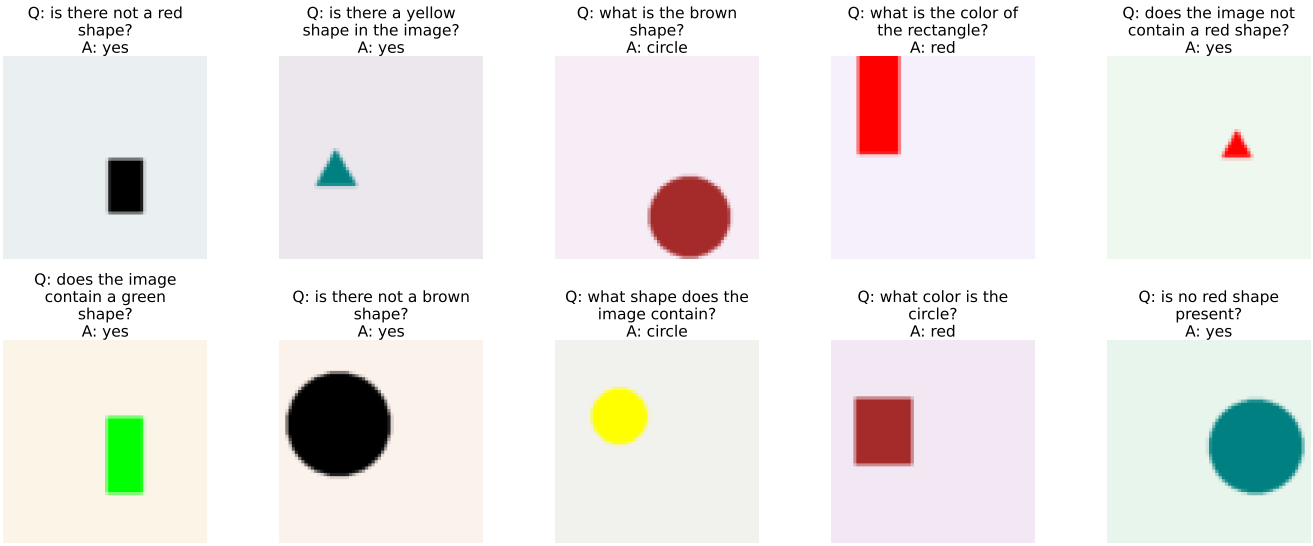


Figure 9. Examples of questions and answers from the Easy-VQA dataset.

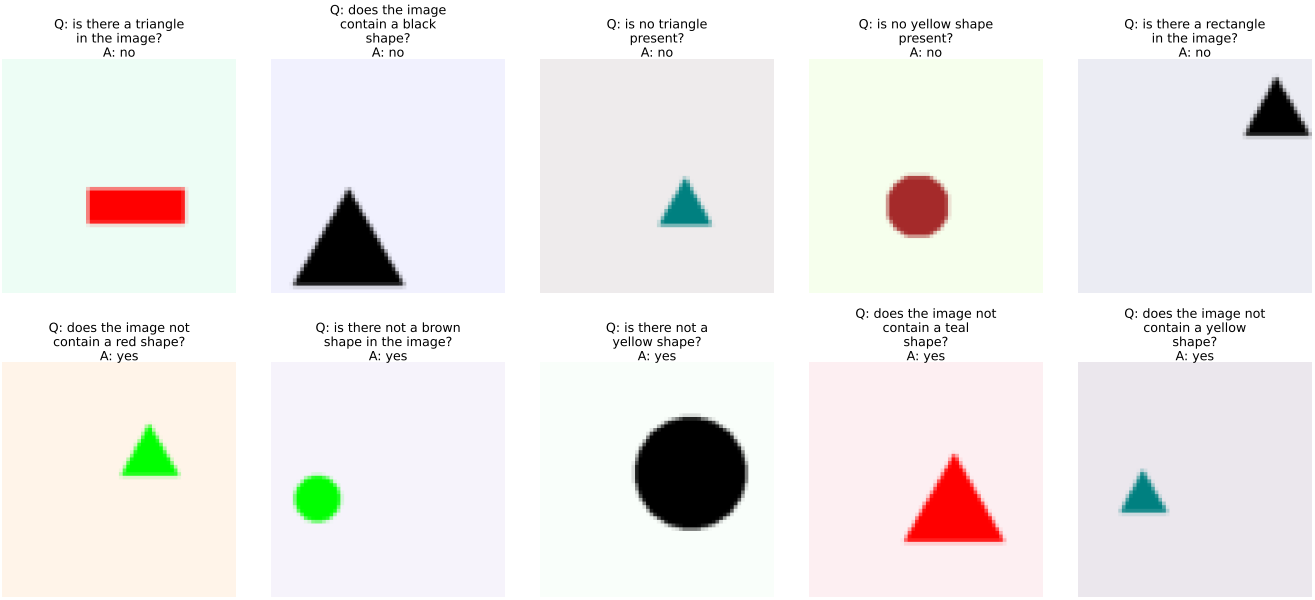


Figure 10. Challenging classes (yes/no) from the EasyVQA dataset

Figure 11. Examples of questions and answers from the Daquar dataset.

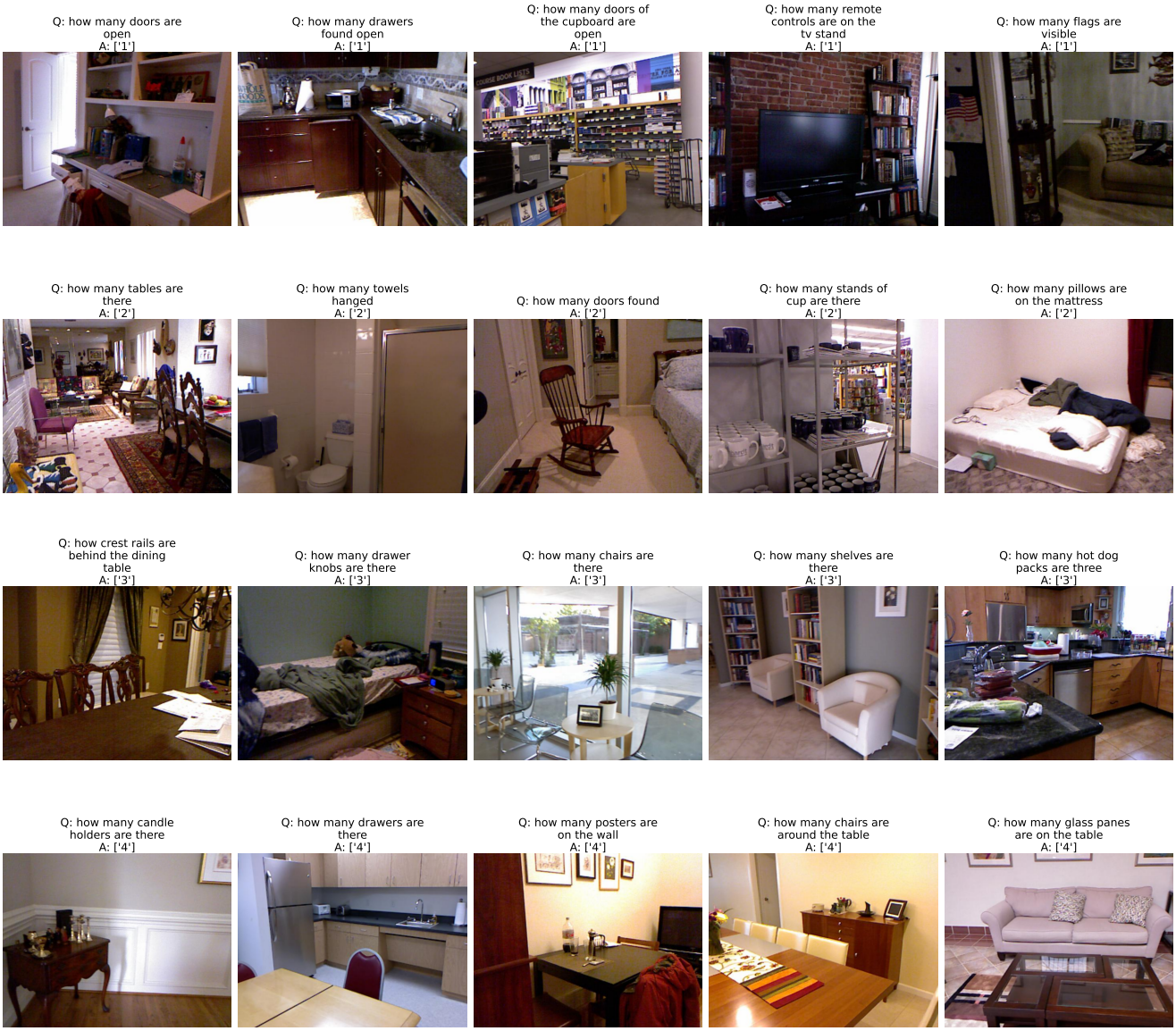


Figure 12. Challenging classes (digits) from the Daquar dataset.