Supplementary Material

6. Additional Analysis

We provide additional analysis of our benchmark and model performance.

6.1. Album Size Distribution

We plotted the distributions of album sizes in the training set, the test set, and all combined in Fig. 4. The majority of the albums consist of about 30 to 40 images. The average number of images per album is about 42.

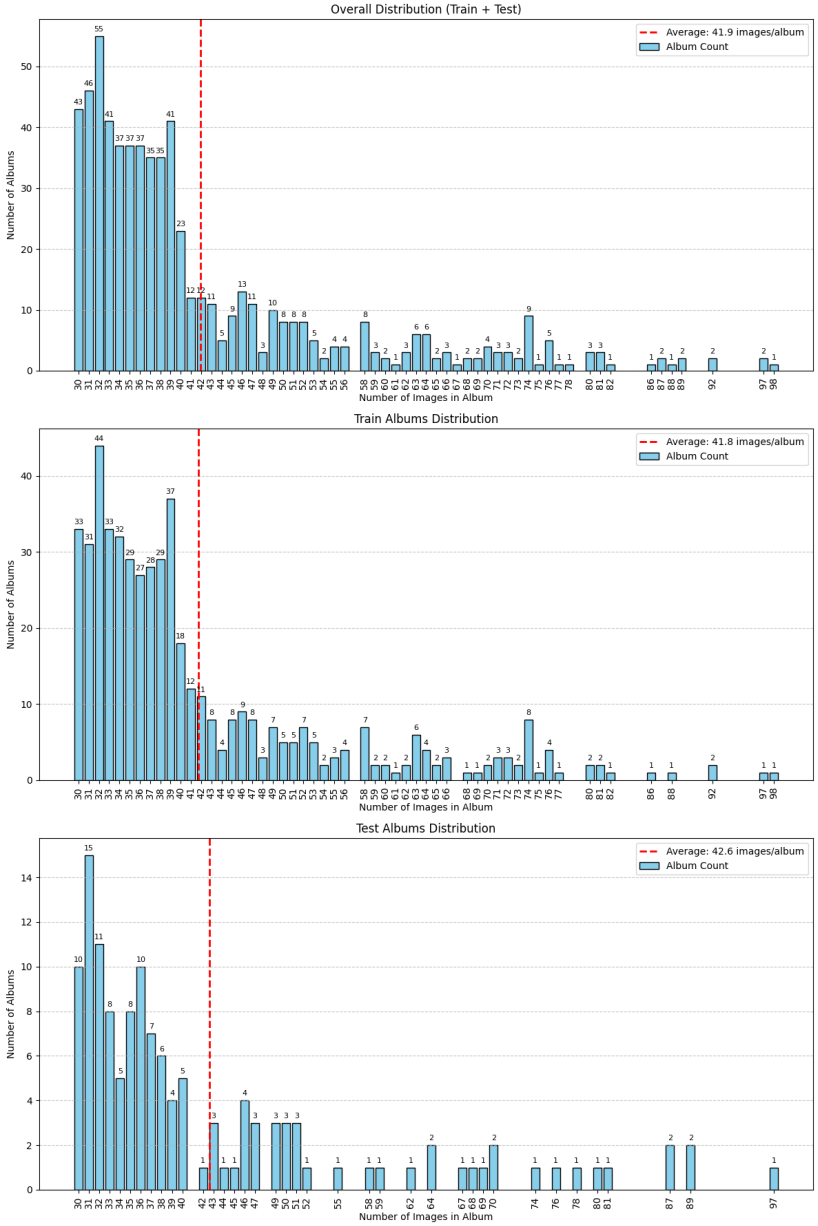


Figure 4. Distributions of image numbers per album.

6.2. Position Bias in Image Selection

Position bias has proven to be a prevalent issue of modern transformer-based models where they prioritize content based on its position within the given context[2]. It has been shown that position bias can hurt models' performance and even cause model failure in scenarios like LLM-as-a-Judge[3], recommendation tasks[4], and multi-image Question Answering[5].

To examine whether the model exhibits positional bias when selecting images from albums, we plotted the distribution of selected images' relative positions in Intent Selection tasks. The distribution of selected images in the ground truth annotations is illustrated in Fig. 5. We see that images are selected fairly evenly across different positions, showing no bias towards any particular position within the albums. In Fig. 6, we show that when language context is provided, models' image selections follow the distribution pattern of ground truth annotations. This is expected because when language context is given, an album is processed by the model one image at a time, which gives the model no extra information about the relative position of each image within that album.

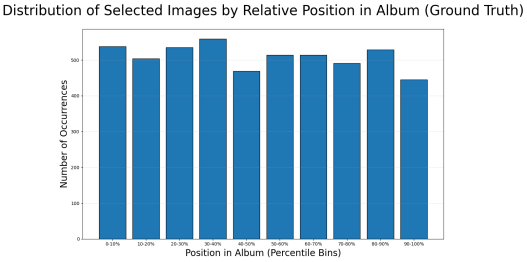


Figure 5. Distributions of selected image positions in Intent Selection task (ground truth). X-axis: bins of image positions. Y-axis: number of occurrences.

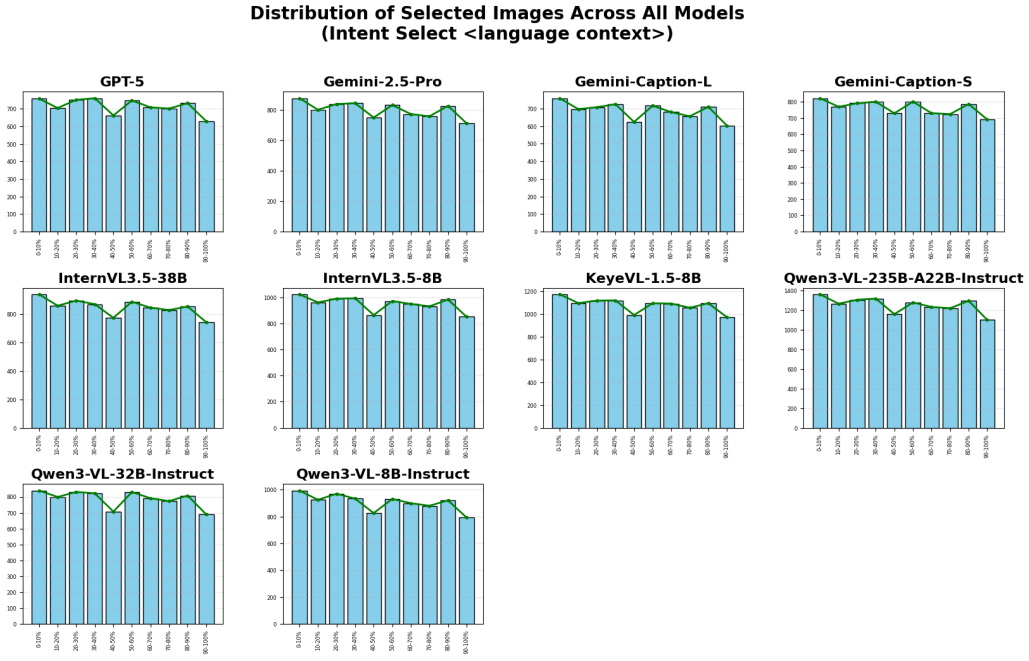


Figure 6. Distributions of selected image positions in Intent Selection task with language context. X-axis: bins of image positions. Y-axis: number of occurrences.

[2]Eliminating Position Bias of Language Models: A Mechanistic Approach. Ziqi Wang, et al. https://openreview.net/pdf?id=fvkElsJOsN

[3]Judging the Judges: A Systematic Investigation of Position Bias in Pairwise Comparative Assessments by LLMs. Lin Shi, et al. https://arxiv.org/abs/2406.07791v1

[4]Evaluating Position Bias in Large Language Model Recommendations. Ehtan Bito, et al. https://arxiv.org/abs/2508.02020

[5]Identifying and Mitigating Position Bias of Multi-image Vision-Language Models. Xinyu Tian, et al. CVPR, 2025.

On the other hand, when visual context is provided instead of language context, models are exposed to the whole album all at once, making them more vulnerable to position bias. For example, in Fig. 7, InternVL3.5-38B, -VL-1.5-8B, Qwen3-VL-32B-Instruct, and Qwen3-VL-8B-Instruct present a clear ascending trend in their image selections, which means they are biased towards images that appear near the end of the album. On the contrary, all the Gemini models show a clear descending trend, which means they are slightly biased toward images that appear at the beginning of the album. Models are clearly more biased when visual context (the whole album) is provided comparing with language context (one image at a time). InternVL3.5-8B shows an interesting pattern where it tends to be biased toward images that appear at around 70% into the album.

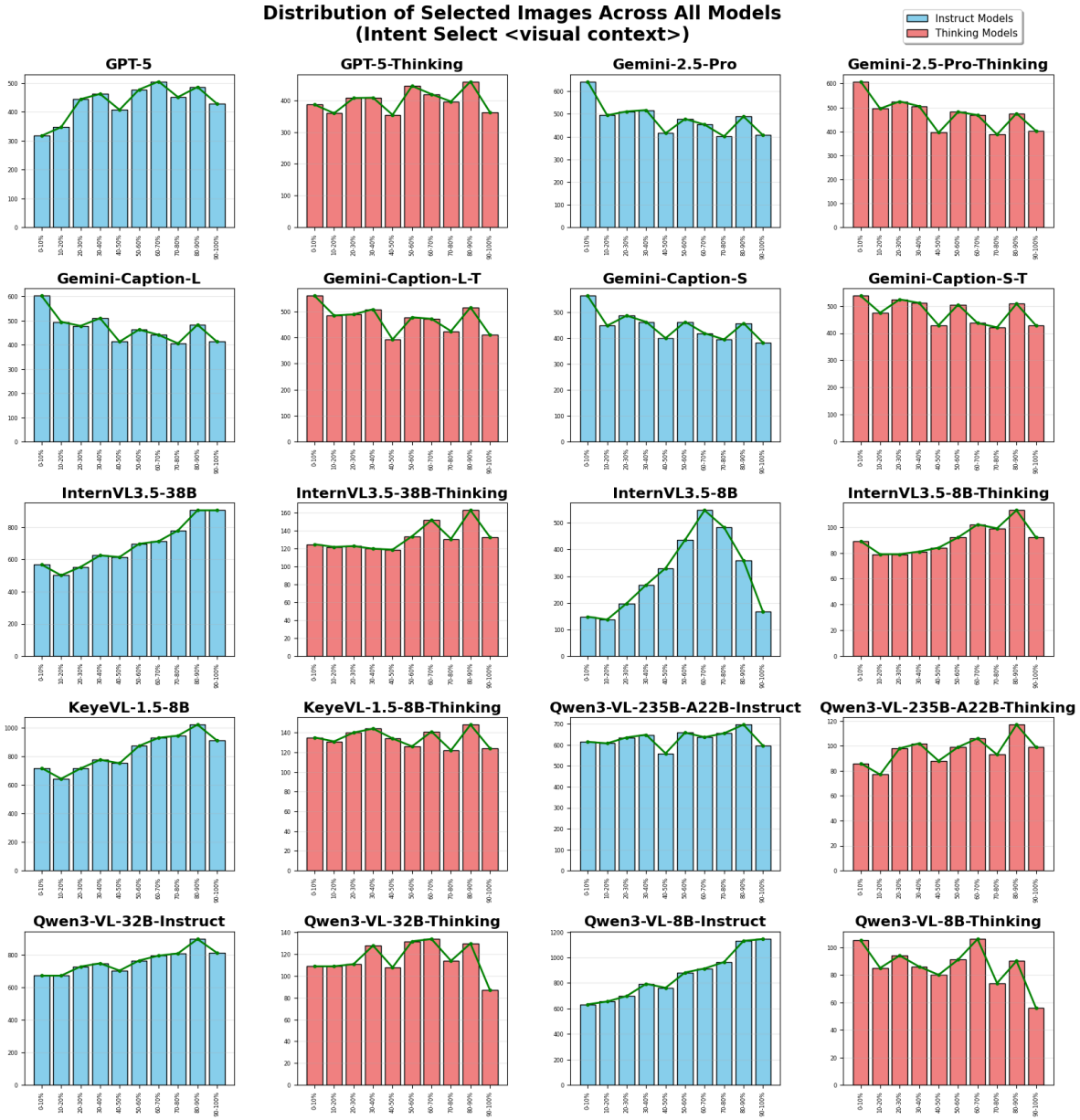


Figure 7. Distributions of selected image positions in Intent Selection task with visual context. X-axis: bins of image positions. Y-axis: number of occurrences.

6.3. Performance of CLIP-like Vision Language Models

We also test how CLIP-like VLMs would perform on our tasks. We choose SigLIP-2 (So400m-patch14-384) for its excellent image classification and retrieval capabilities. We evaluate it on Intent Selection and Intent Rating tasks only. Since SigLIP-2 is a vision encoder and does not generate natural language, we compute the similarity scores between its generated image embeddings and text embeddings (task query). For Intent Selection, we treat the top-k images as the model's image selection regarding a given query. We set k to be equal to the number of images in the ground truth selections for each album. For Intent Rating, we sort and group the image embeddings evenly into 4 bins based on their similarity scores with the encoded task query, representing the model giving a score of 0 to 3, respectively.

For Intent Selection, SigLIP-2 achieves an f1-score of 0.281 and mAP of 0.352. For Intent Rating it has an accuracy of 0.186, MAE of 1.426, and RMSE of 1.693. The scores are worse than all other methods in Table 1.

6.4. Raw Parsing Failure Rate

Table 4. Raw parsing-failure rate (visual context)

Model	Selection	Rating	Group Clustering	Group Labeling
GPT-5	0.000%	0.256%	0.000%	0.000%
GPT-5-Thinking	0.000%	0.000%	0.000%	0.000%
Gemini-2.5-Pro	0.000%	0.000%	0.000%	0.000%
Gemini-2.5-Pro-Thinking	0.000%	0.000%	0.000%	0.000%
Keye-VL-1.5-8B	0.000%	0.000%	3.788%	0.000%
Keye-VL-1.5-8B-Thinking	0.000%	0.000%	34.783%	11.594%
InternVL3.5-8B	0.000%	0.256%	0.388%	0.775%
InternVL3.5-8B-Thinking	0.000%	0.000%	11.594%	2.899%
Qwen3-VL-8B-Instruct	0.000%	0.000%	0.000%	0.000%
Qwen3-VL-8B-Thinking	0.000%	8.571%	52.174%	21.739%
Qwen3-VL-32B-Instruct	0.000%	0.000%	0.000%	0.000%
Qwen3-VL-32B-Thinking	0.000%	0.952%	49.275%	28.985%
InternVL3.5-38B	0.000%	0.000%	0.000%	0.000%
InternVL3.5-38B-Thinking	0.000%	0.000%	2.899%	1.449%
Qwen3-VL-235B-A22B-Instruct	0.000%	0.000%	0.000%	0.000%
Qwen3-VL-235B-A22B-Thinking	1.905%	0.000%	4.348%	4.348%
Gemini-Caption-Long	0.000%	0.000%	0.000%	0.000%
Gemini-Caption-Short	0.501%	0.000%	0.000%	0.000%
Gemini-Caption-Long-Thinking	0.256%	0.000%	0.775%	0.000%
Gemini-Caption-Short-Thinking	1.253%	0.000%	0.000%	0.000%

Table 5. Raw parsing-failure rate (language context)

Model	Selection	Rating	Group Labeling
GPT-5	1.253%	0.000%	0.000%
Gemini-2.5-Pro	1.026%	0.000%	0.000%
Keye-VL-1.5-8B	0.251%	0.000%	0.000%
InternVL3.5-8B	0.251%	0.000%	0.000%
Qwen3-VL-8B-Instruct	0.501%	0.000%	0.000%
Qwen3-VL-32B-Instruct	1.253%	0.000%	0.000%
InternVL3.5-38B	0.501%	0.000%	0.000%
Qwen3-VL-235B-A22B-Instruct	0.758%	0.000%	0.000%
Gemini-Caption-Long	1.609%	0.000%	0.000%
Gemini-Caption-Short	1.018%	0.000%	0.000%

Parsing errors were found in 12 of the 20 model configurations shown in Table 1 of the paper when naive parsing was used instead of Gemini-2.5-Flash. Table 4 and Table 5 show the raw parsing failure rates for each model configuration on each task type with visual context and language context, respectively.

Our manual audit showed that Gemini-2.5-Flash did not alter the semantic content. We randomly sampled ∼10% of the albums and found no semantic changes in the results. Note that our post-process had no access to the images so it could not make its own interpretation of the data.

6.5. Cost/Performance Trade-offs between Instruct and Thinking Models

We show below in Table 6 and Table 7 the compute time (in seconds) used by each model configuration to complete each task type, averaged across all albums.

Table 6. Average compute time (seconds) with visual context

Model	Selection	Rating	Group Clustering	Group Labeling
GPT-5	15.18	18.62	18.48	17.99
GPT-5-Thinking	49.06	63.59	83.88	73.72
Gemini-2.5-Pro	5.31	6.67	6.36	5.99
Gemini-2.5-Pro-Thinking	22.06	26.66	32.86	27.99
Gemini-Caption-Long	3.63	5.54	5.45	4.98
Gemini-Caption-Long-Thinking	22.01	25.65	27.08	25.55
Gemini-Caption-Short	3.42	5.24	4.93	4.77
Gemini-Caption-Short-Thinking	20.09	23.09	28.35	24.28
InternVL3.5-38B	27.20	58.55	48.64	46.16
InternVL3.5-38B-Thinking	102.26	187.70	223.63	218.04
InternVL3.5-8B	6.14	26.69	24.57	21.87
InternVL3.5-8B-Thinking	83.61	109.94	248.36	226.04
Keye-VL-1.5-8B	7.04	19.23	22.02	17.68
Keye-VL-1.5-8B-Thinking	101.08	124.96	181.40	164.31
Qwen3-VL-235B-A22B-Instruct	6.06	12.81	10.56	10.36
Qwen3-VL-235B-A22B-Thinking	93.70	81.24	131.25	123.32
Qwen3-VL-32B-Instruct	15.24	45.24	83.36	41.69
Qwen3-VL-32B-Thinking	305.69	283.34	440.42	391.53
Qwen3-VL-8B-Instruct	8.65	23.79	33.01	28.88
Qwen3-VL-8B-Thinking	190.98	225.12	266.35	239.12

7. Impact on Visual Context for Query Selection

Our task could be considered as a type of image retrieval. In general, many image retrieval prompts can be answered without requiring the context of an album (e.g. Find all images with a dog in them). Although this is true for some of the prompts in our test set, many other prompts require or benefit from album-level context.

Fig. 8 shows some examples. For the first row, it is unclear whether the first two images show the full scale of the protest until the next two images are seen. For the second row, it may not be clear for many that the first and even second images demonstrate wedding customs, but further context helps to make that clear.

Table 7. Average compute time (seconds) with language context

Model	Selection	Rating	Group Labeling
GPT-5	121.49	120.97	120.73
Gemini-2.5-Pro	114.55	106.82	110.72
Gemini-Caption-Long	96.40	100.21	100.72
Gemini-Caption-Short	93.39	98.39	95.91
InternVL3.5-38B	35.12	41.31	42.47
InternVL3.5-8B	14.57	17.72	18.55
Keye-VL-1.5-8B	11.69	14.13	14.64
Qwen3-VL-235B-A22B-Instruct	95.94	128.69	128.86
Qwen3-VL-32B-Instruct	26.85	33.04	34.15
Qwen3-VL-8B-Instruct	9.04	16.21	17.94

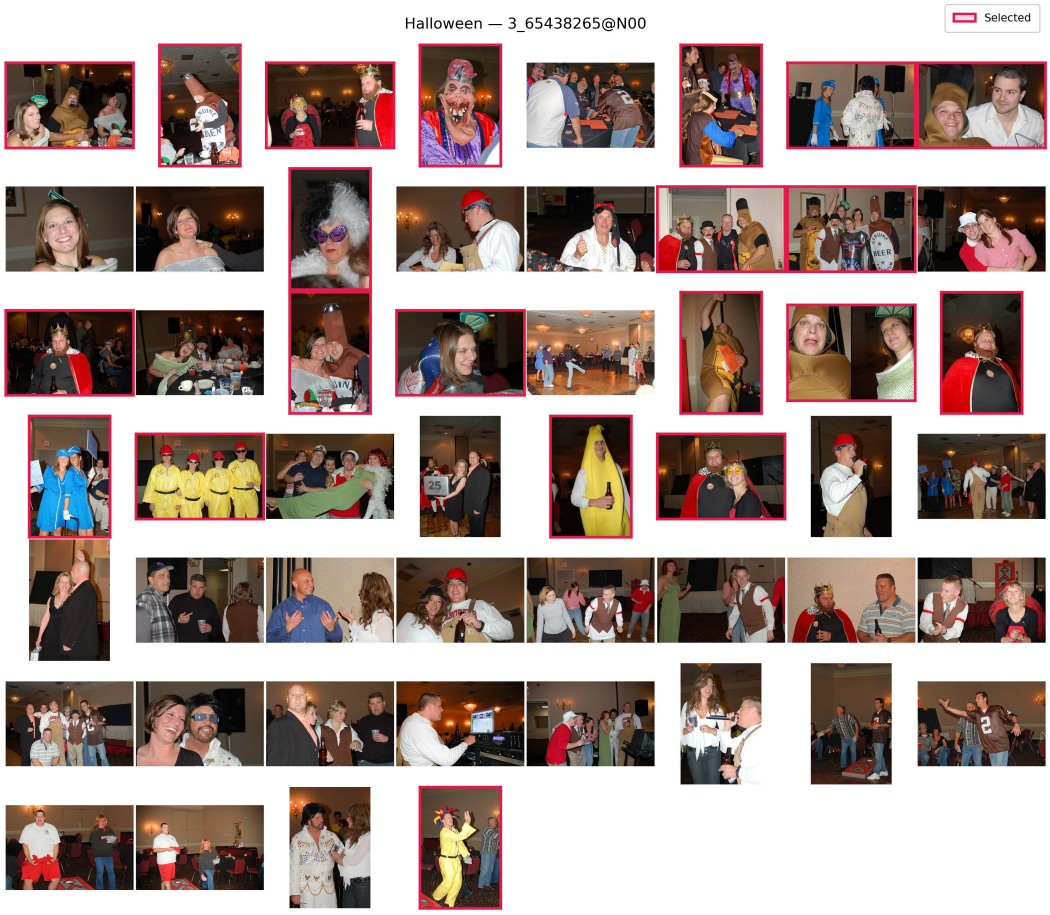
Test set query: Documenting the scale of public participation in protests.

Test set query: Capture the essence of traditional wedding customs and rituals for a cultural study paper.

Figure 8

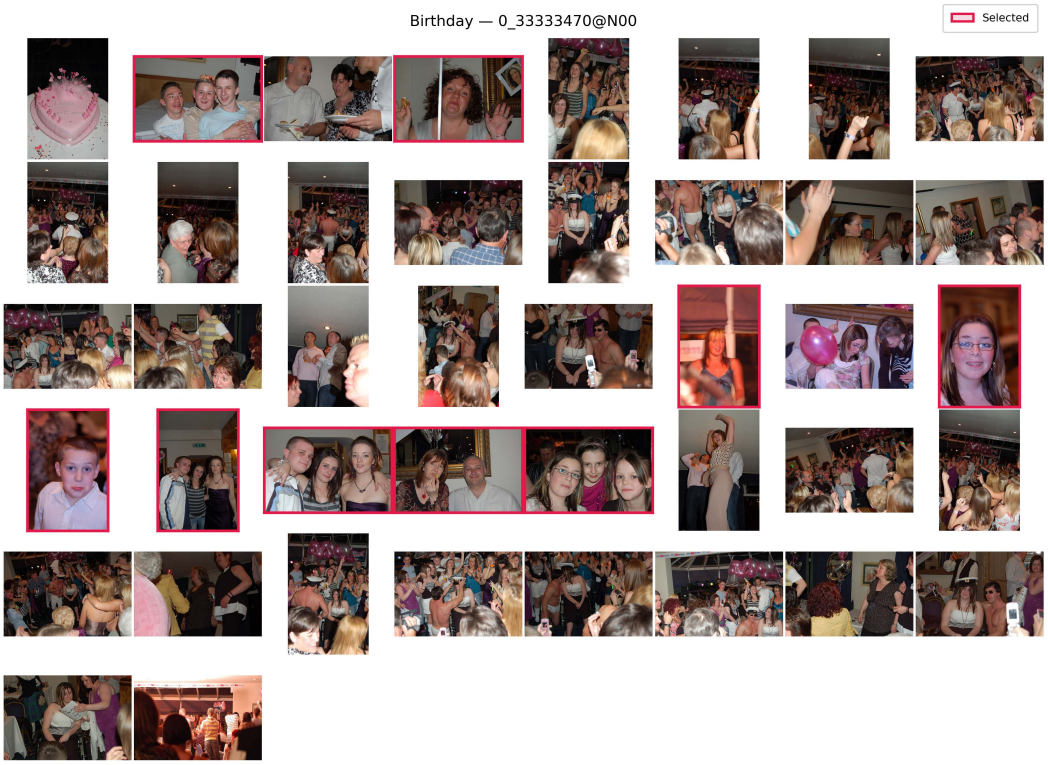
8. Additional Album Examples

We provide several albums with their queries and annotations as additional examples in the following figures.



Query: "Create a guide on how to choose a unique costume for themed parties."

Figure 9

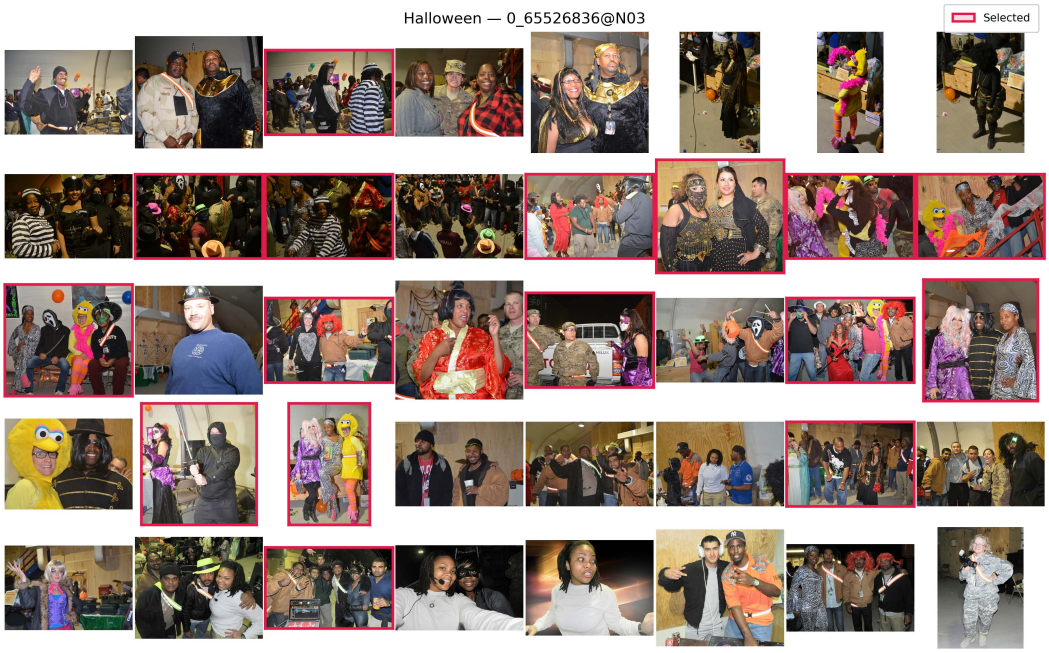


Query: "Select images for a photo album that highlights portraits of guests."

Figure 10

Query: "Documenting the scale of public participation in protests."

Figure 11

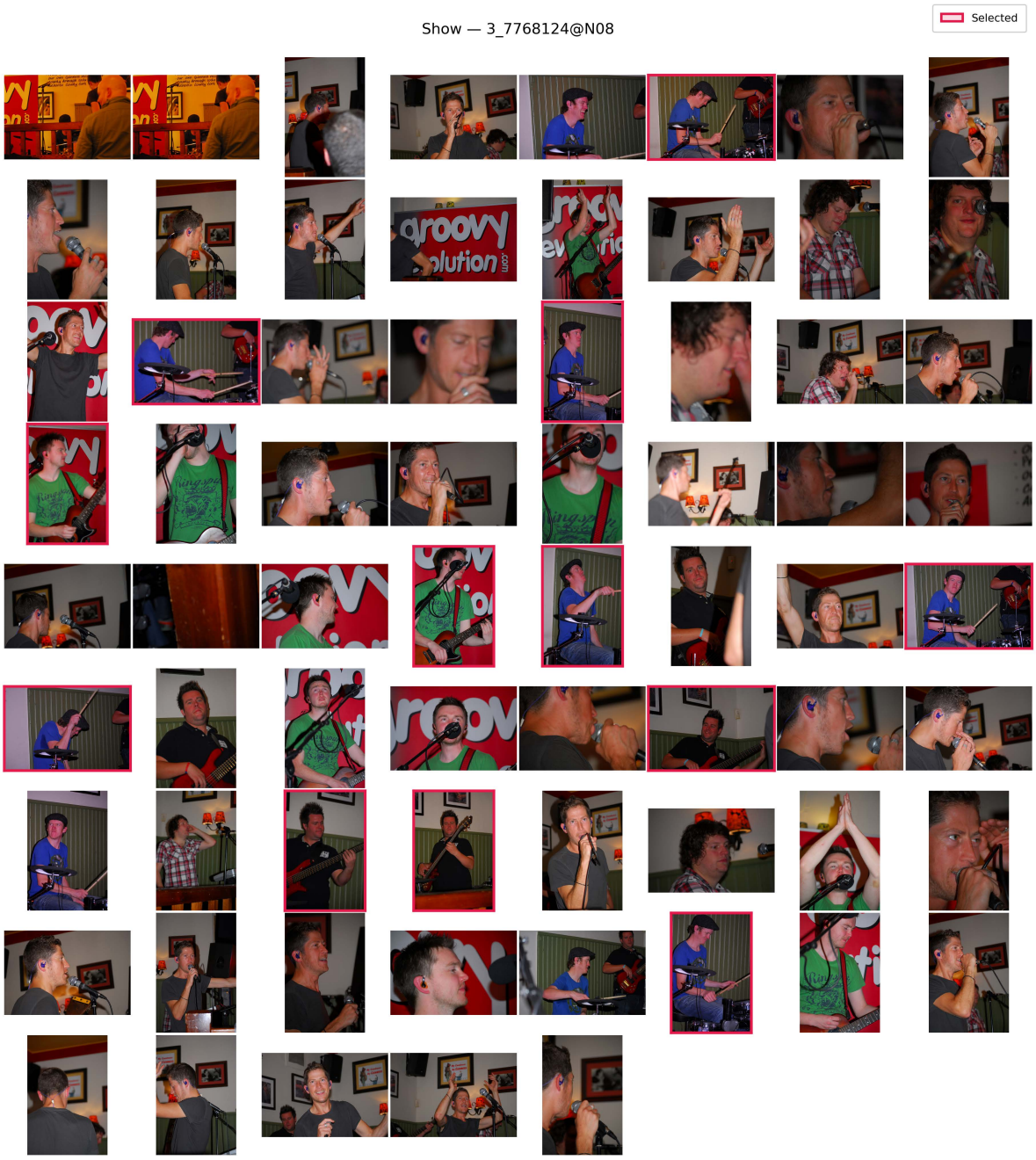


Query: "Document the variety of costumes worn at a theme party."

Figure 12

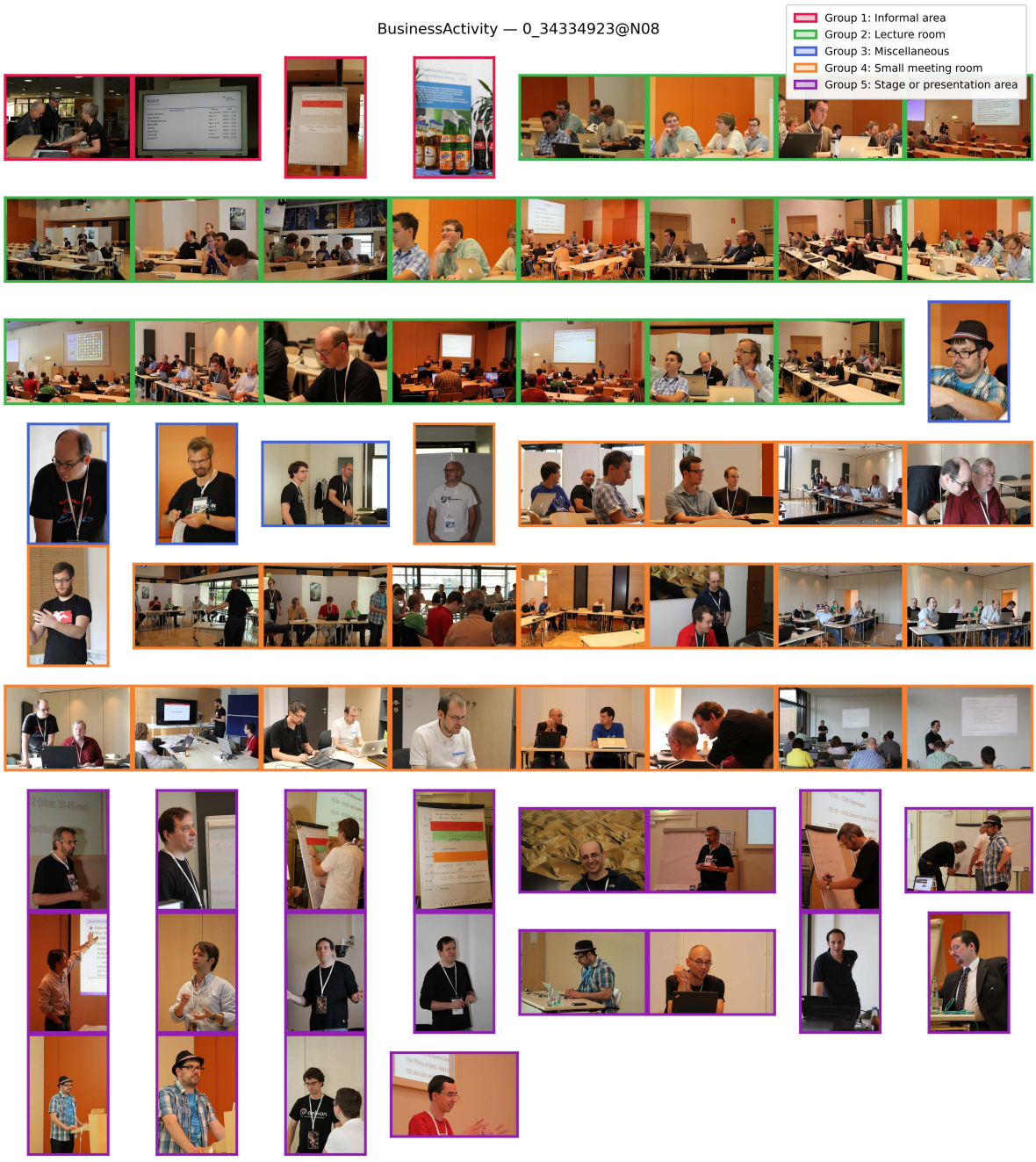
Query: "Gather reference images of fishes to show the ocean biodiversity."

Figure 13



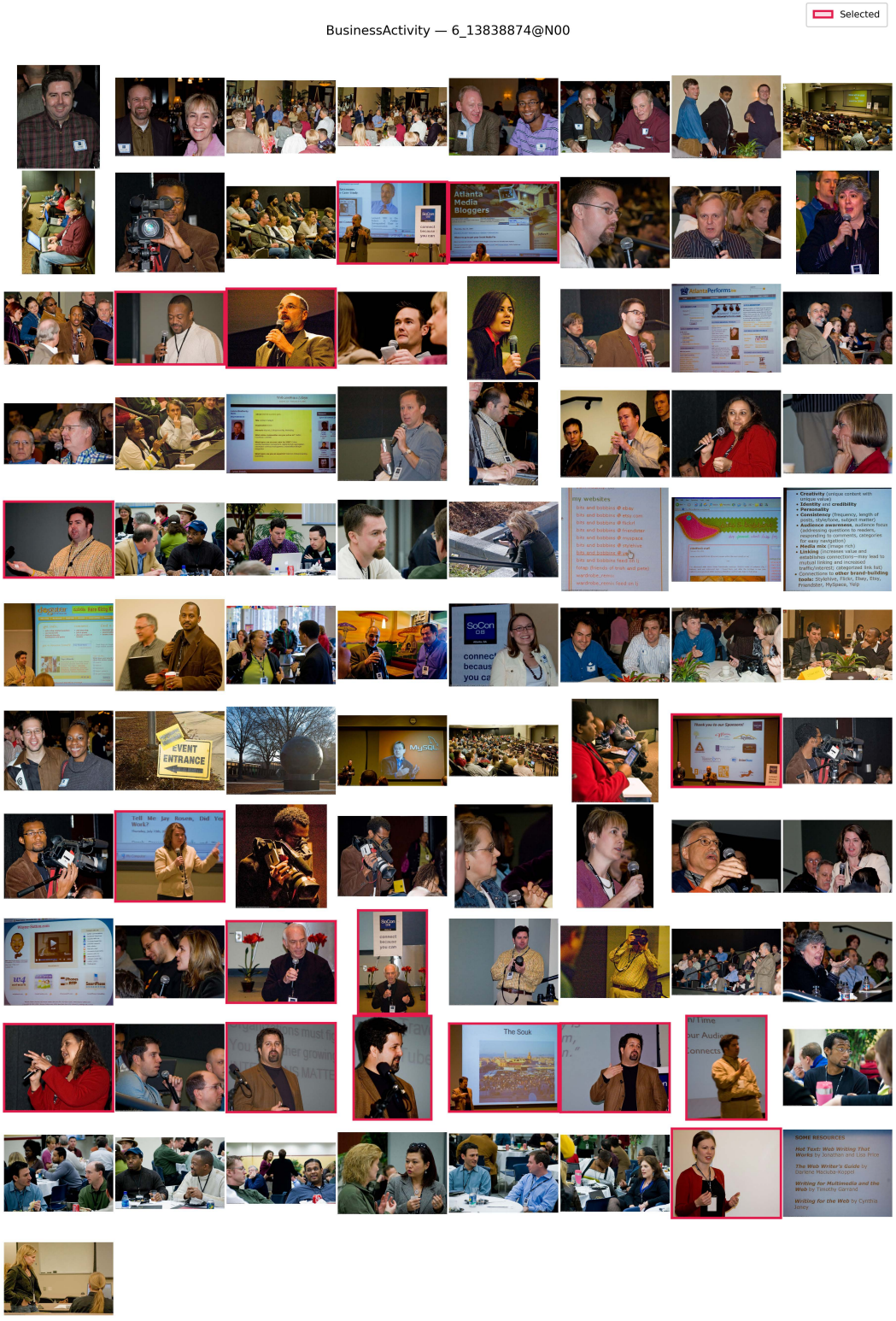
Query: "Document the musical performance by capturing key moments of musicians actively playing their instruments."

Figure 14



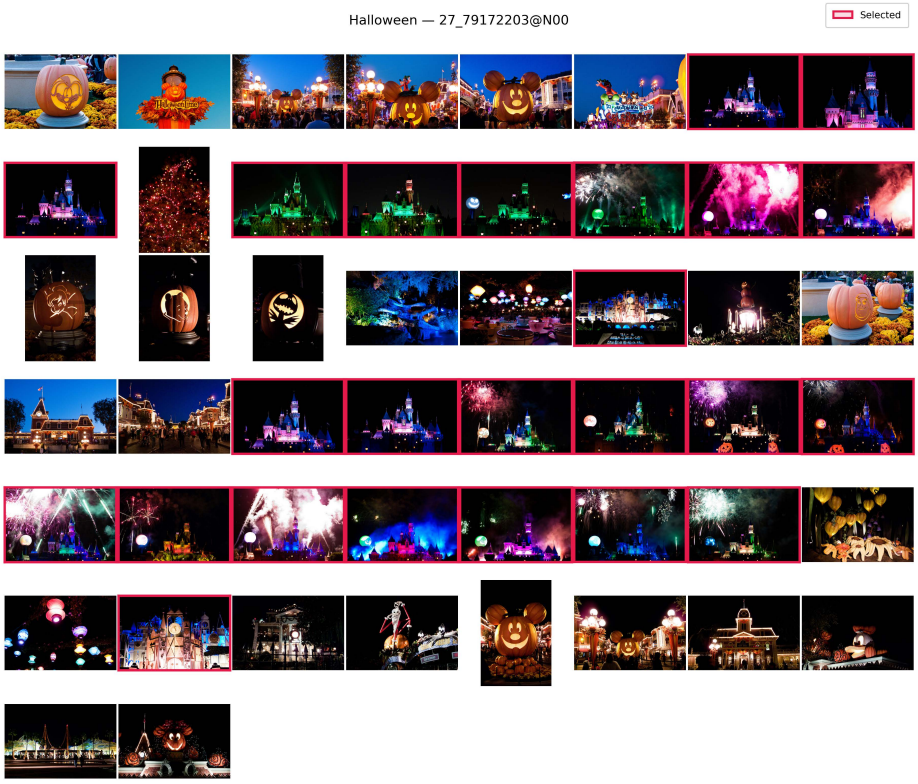
Query: "Location setting within the venue."

Figure 15



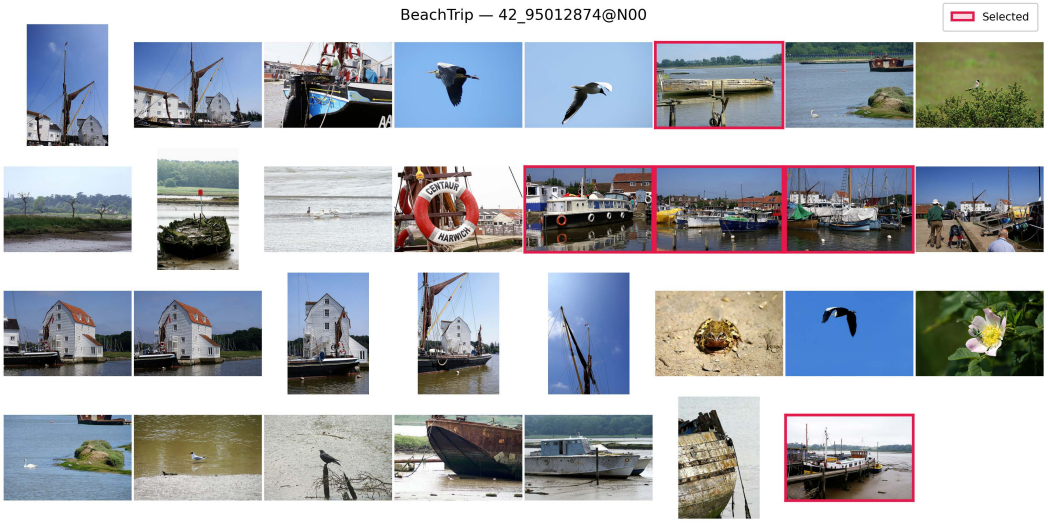
Query: "Identify key speakers during the event for a conference report."

Figure 16



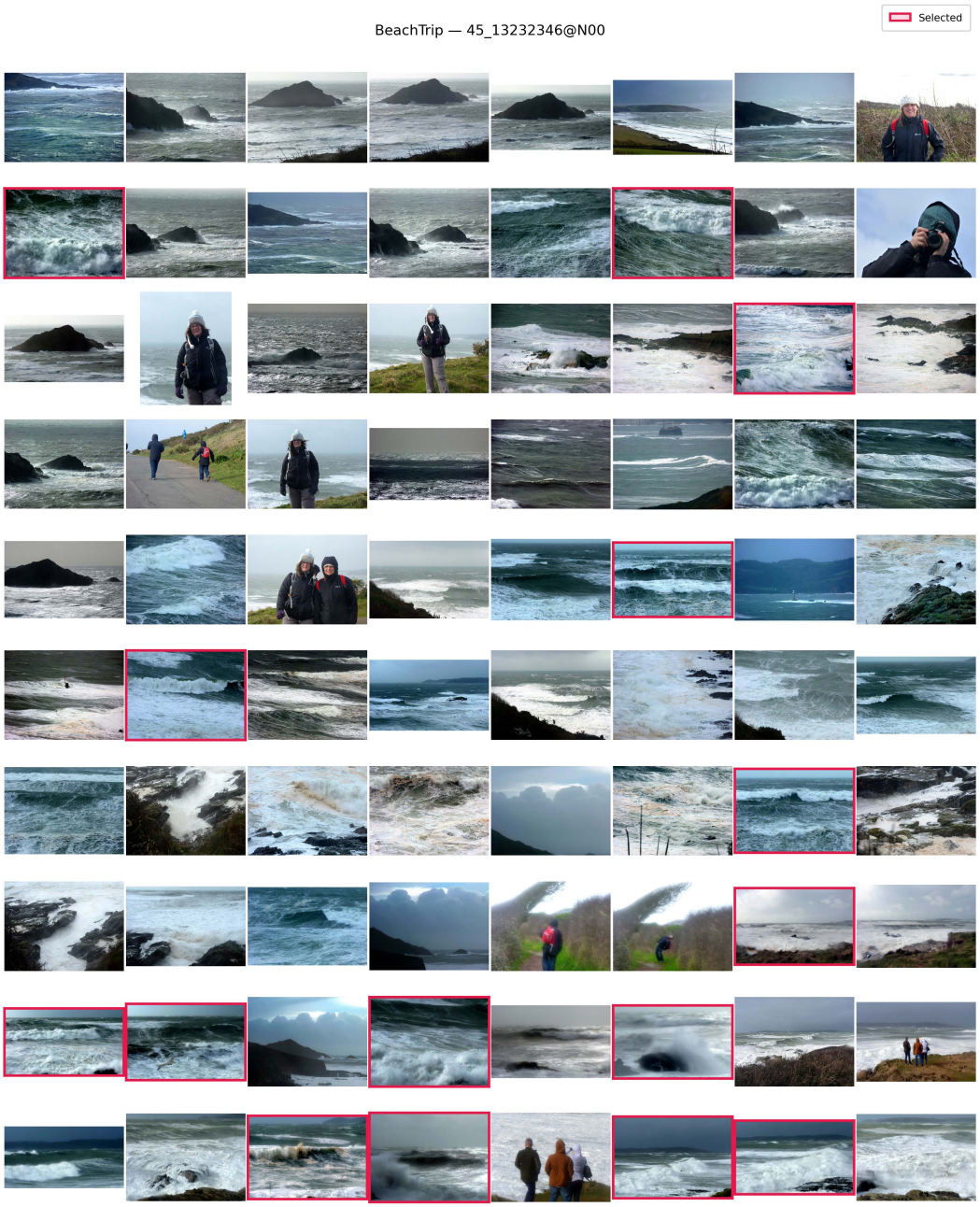
Query: "Compile a visual collection for a travel blog post highlighting the architectural beauty of themed attractions at night."

Figure 17



Query: "Assemble a collection of images highlighting different types of boats and watercraft in the harbor."

Figure 18



Query: "I want to select images that showcase the most turbulent ocean waves to create a dramatic travel blog about extreme weather conditions."

Figure 19

9. Qualitative Analysis

We provide a sample album and different model outputs as a case study. The original album is shown in Fig. 20.

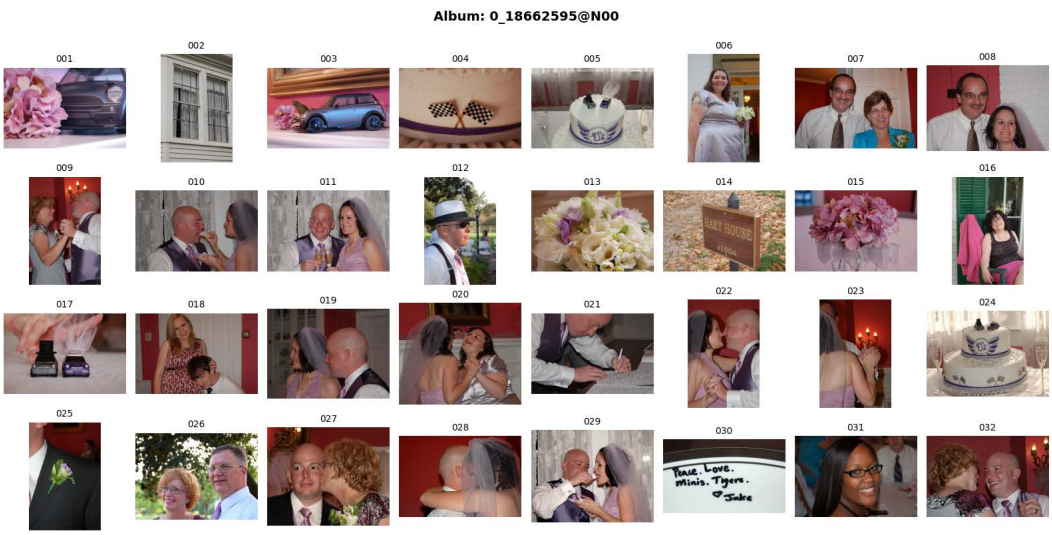


Figure 20. An album of a wedding ceremony. Album ID: 0_18662595@N00. Number of images: 32.

9.1. Intent Selection (with visual context)

The ground truth annotation of this task is shown in Fig 21. The outputs of all instruct models are shown in Fig 22; the outputs of all thinking models are shown in Fig 23.

Upon closer inspection of each model's output, we find several interesting patterns:

Models tend to select more images than needed. We find that models tend to select more images than necessary, which increases their chance of getting all the correct images while also increasing the false positive rate. We see the same pattern in Tables 1 and 2, under Intent Selection task, where models usually have higher recall scores than precision scores. One extreme example is Qwen3-VL-235B-A22B-Instruct in Fig. 22, where it selects 17 out of 32 images.

Models demonstrate common sense. There are certain images in this album that no models picked or that nearly all of the models picked, which means models share common knowledge in perceiving the subject of the album. For example, Image-022, where the couple is kissing, is consistently selected by models (except InternVL3.5-8B-Thinking) to represent one of the key moments of the wedding ceremony. On the other hand, images like 001 (decoration), 002 (outdoor setting), 026 (wedding guests) were never selected by any of the models.

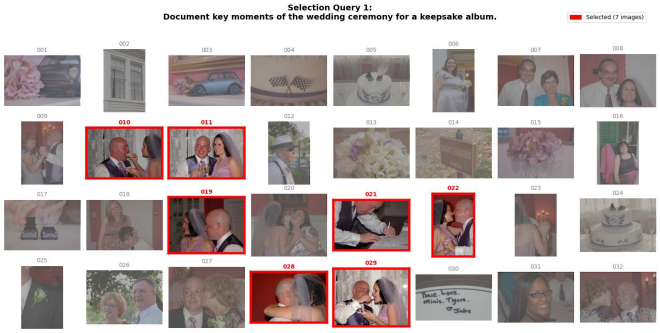


Figure 21. The ground truth annotation of Intent Selection query 1: "Document key moments of the wedding ceremony for a keepsake album."

Figure 22. All instruct models' outputs visualized. Selected images are highlighted.

Figure 23. All thinking models' outputs visualized. Selected images are highlighted.

9.2. Intent Rating (with visual context)

The ground truth annotation of this task is shown in Fig 24. The outputs of all instruct models are shown in Fig 25; the outputs of all thinking models are shown in Fig 26.

By looking closely at each model's output, we find these interesting patterns:

Models agree on high-rating images. Image-004, Image-005, and Image-024 are the only images that have the highest rating in ground truth (Fig. 24). Looking at the models' outputs, they are consistently rated as 3 with several exceptions where they are rated as 2. This means that models agree on what these images are about and why they are semantically highly related to the query content.

Thinking models tend to give images higher ratings. We find that thinking models usually have more images rated as 1; while instruct models tend to rate images 0 when possible. The two obvious examples are InternVL3.5-38B-Thinking and Qwen3-VL-32B-Thinking, as shown in Fig. 26, they give the majority of the album a score 1, whereas in Fig. 25, their instruct counterparts rate the majority of the album as 0.

Figure 24. The ground truth annotation of Intent Rating query 2: "Gather inspiration images for a themed wedding cake design." Red: images with a rating of 3; Orange: images with a rating of 2; Blue: images with a rating of 1; Gray: images with a rating of 0.

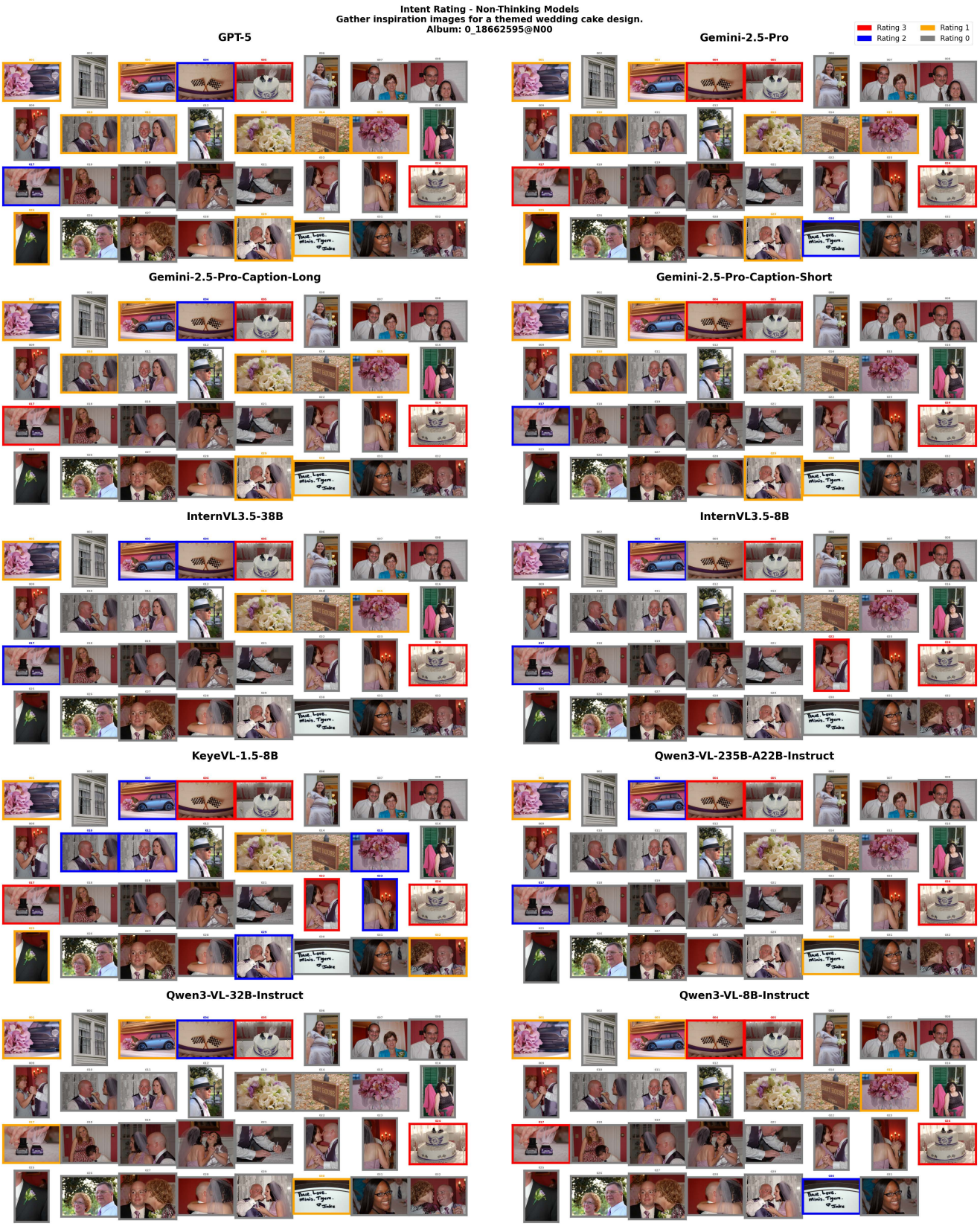


Figure 25. All instruct models' outputs visualized. Different ratings are colored differently.

Figure 26. All thinking models' outputs visualized. Different ratings are colored differently.

9.3. Group Labeling and Group Clustering (both with visual context)

The ground truth annotations for these two tasks are shown in Fig. 27. For the Group Labeling task, the outputs of all instruct models are shown in Fig. 28; the outputs of all thinking models are shown in Fig. 29. For the Group Clustering task, the outputs of all instruct models are shown in Fig. 30. The outputs of all thinking models are shown in Fig. 31.

Looking closely at each model's output, we identified the following patterns:

Proprietary models are more robust compared to open-source ones. We find that closed-source models almost always perfectly follow the task instructions while open-sourced models often fail to do so. One example is images getting left out in predicted groupings. In Fig. 28, Fig. 29, Fig. 30, and Fig. 31, we constantly see closed-source models (the top 2 rows in each plot) perfectly assigns every image to a group, with the only exception in Fig. 30 where GPT-5 failed to include Image-021 in any of the groups. On the other hand, open-sourced models often fail to include all the images. One extreme example is InternVL3.5-8B in Fig. 30 where it missed 8 images in total. We give further failure mode analysis in Sec. 10.

When categories are not given, models can still propose reasonable partitions. We consider Group Clustering a harder task than Group Labeling in the sense that models must come up with reasonable categories to partition the images on their own, based on the grouping query. This difference in difficulty is also reflected in Table 1, where models usually achieve higher ARI and NMI scores (metrics that measure clustering alignment) under Group Labeling tasks than under Group Clustering tasks. However, we find that, in general, when categories are not provided to the models, they can still generate reasonable partitions even when not perfectly aligned with the ground truth categories. For example, as shown in Fig. 30 and Fig. 31, we see that although the ground truth categories are not provided to the models, many of them can still generate semantically meaningful and reasonable categories within a wedding ceremony context. Many predicted categories cover subjects such as "Decorations," "People"/"Guests"/"Portraits," and "Candid Moments"/"Interactions." Although not perfectly aligned with the ground truth categories, the model-generated categories usually cover a similar set of concepts. One future direction would be to account for different but reasonable partitions in the evaluation process.

Figure 27. The ground truth annotation of Grouping query 1: "Type of Scene." Different colors represent different groups.

Figure 28. Group Labeling task. All instruct models' outputs visualized. Images of the same group share the same color.

Figure 29. Group Labeling task. All thinking models' outputs visualized. Images of the same group share the same color.

Figure 30. Group Clustering task. All instruct models' outputs visualized. Images of the same group share the same color.

Figure 31. Group Clustering task. All thinking models' outputs visualized. Images of the same group share the same color.

10. Failure Modes

Models mainly fail at the Group Clustering and Group Labeling tasks.

Missing images and overlapping groups. As we discussed in Sec. 4, models sometimes may leave images out of any of the groupings and may generate overlapping groupings. For example, as shown in Fig. 28, InternVL3.5-38B, Keye-VL-1.5-8B, and InternVL3.5-8B, all missed an image or two in their groupings. In the same figure we also show that Qwen3-VL-8B-Instruct generates severely overlapped groups, with the actual model output being the following:

```
Group-1: Detail and Decoration
[Image-1, Image-2, Image-3, Image-4, Image-5, Image-13, Image-14, Image-15, Image-17,
    Image-20, Image-25, Image-30]

Group-2: Couple Portraits
[Image-6, Image-7, Image-8, Image-9, Image-10, Image-11, Image-12, Image-16, Image-18,
     Image-19, Image-21, Image-22, Image-23, Image-24, Image-26, Image-27, Image-28,
    Image-29]

Group-3: Guest Interactions
[Image-1, Image-3, Image-5, Image-6, Image-7, Image-8, Image-9, Image-10, Image-11,
    Image-12, Image-13, Image-14, Image-15, Image-16, Image-17, Image-18, Image-19,
    Image-20, Image-21, Image-22, Image-23, Image-24, Image-25, Image-26, Image-27,
    Image-28, Image-29, Image-30, Image-31]

Group-4: Guests Portraits
[Image-1, Image-3, Image-5, Image-6, Image-7, Image-8, Image-9, Image-10, Image-11,
    Image-12, Image-13, Image-14, Image-15, Image-16, Image-17, Image-18, Image-19,
    Image-20, Image-21, Image-22, Image-23, Image-24, Image-25, Image-26, Image-27,
    Image-28, Image-29, Image-30, Image-31]

Group-5: miscellaneous
[Image-1, Image-2, Image-3, Image-4, Image-5, Image-6, Image-7, Image-8, Image-9,
    Image-10, Image-11, Image-12, Image-13, Image-14, Image-15, Image-16, Image-17,
    Image-18, Image-19, Image-20, Image-21, Image-22, Image-23, Image-24, Image-25,
    Image-26, Image-27, Image-28, Image-29, Image-30, Image-31]
```

which clearly shows duplicate group assignments to many images in the album.

Model gives empty answers. Models will occasionally ignore all the instructions and give empty answers. As shown in Fig. 29, Qwen3-VL-235B-A22B-Thinking did not generate any valid groupings, which results in all the images being classified as "misc" by default.

Due to unknown reasons, Gemini models will consistently refuse to answer any question related to several specific albums. We think the reason might be that certain images in those albums triggered Gemini's safety concerns, which causes Gemini not to answer any question at all.

Over-simplified categories and conceptually overlapped categories. When grouping categories are not provided, models are asked to come up with a list of categories that best partition the album without overlapping concepts, conditioned on a grouping query. However, we find that sometimes the model will generate over-simplified categories and conceptually overlapped categories. For example, in Fig. 31, Keye-VL-1.5-8B-Thinking generates two groups: "Wedding Scenes" and "Misc." Since the whole album is about a wedding ceremony, we consider this generated grouping low-quality and overly simplified.

We find that models sometimes will generate categories that are too generic, which leads to overlap between categories. For example, in Fig. 30, InternVL3.5-8B, InternVL3.5-38B, and Qwen3-VL-32B-Instruct all give a category named "People," which is too generic to be useful, and overlaps with wedding-specific categories like "Wedding Ceremony".

11. Prompt Templates

As mentioned in 4.1, we needed to form a prompt template was worked well across all VLMs we tested. The prompt needed to be specialized with each method in mind so that a single fair prompt could be used across all methods. In other words, simple naive prompts may not work well for some methods, while extensive prompt engineering may yield unfair results if we put more work into or had more luck in crafting prompts for one method than another, so we instead worked hard prompting all methods to see what was needed to get the best result we could for each method, then combined the needed information into a single prompt that worked well across all methods. While extensive prompt engineering may yield slightly better results for a given method, such work is outside the scope of this paper.

We found it necessary to tailor different prompting strategies to each of the tasks we defined. Here, we provide prompt templates used for each task type and for post-processing. The prompt template used to solve each task comprises two parts: "album context" + "task-specific prompt."

For tasks with visual context, the "album context" part includes all the actual images of the album to provide contextual information:

```
"Here is an album of images:
Image-001: <image>, Image-002: <image>, Image-003: <image>, ..."
```

For tasks with language context, the "album context" part uses album caption instead of actual images:

```
"Here is an album about {caption}.
Here is an image drawn from the album: <image>."
```

The "task-specific prompts" and their corresponding post-processing prompt templates are provided in the following subsections.

11.1. Intent Selection (visual context)

```
"You are a helpful assistant that selects images from an album.
Selection Query: {query}

Your task:
Select images based on the criterion above.

Output format - you MUST follow this exactly:
Image-<id>

Requirements:
- Choose images that BEST match the query (top matches only)
Format Requirements:
- Each line: Image-<id>
- No bullet points, numbering, or extra commentary
- No additional text before or after the selections

Now make your selections:"
```

Post-processing:

```
"Task: Parse the following and convert it to structured JSON format: <model_output>

Extract these exact info:
1. List of image IDs (convert them to 3-digit format like "001", "015")
2. Explanation for each selected image ID (if there is any)

Output JSON sample format:
{{
   "selected_images": ["001", "010", "015"],
   "explanations": ["Explanation for image 001", "Explanation for image 010", "
       Explanation for image 015"] # Set this field to null if there is no explanation
```

```
}}

JSON Output:"
```

11.2. Intent Selection (language context)

```
"You are a helpful assistant that selects images from an album.
Selection Query: {query}

Your task:
Determine if this specific image is highly relevant to the query above.

Output format - you MUST follow this exactly:
Answer: <yes or no>

Requirements:
- Answer with ONLY 'yes' or 'no' (lowercase)
- 'yes' means the image is highly relevant to the query
- 'no' means the image is not highly relevant (including somewhat relevant images)

Format Requirements:
- Single line format: Answer: <yes or no>
- No additional text before or after
- Answer must be either 'yes' or 'no' (lowercase)

Now provide your answer:"
```

Post-processing:

```
"Task: Parse the following and convert it to structured JSON format: <model_output>
The input contains multiple lines, each representing one image's selection result in
    format "Image-X: Answer: yes/no - explanation" or "Image-X: Answer: yes/no".

Extract these exact info:
1. List of image IDs that have "yes" answers (convert them to 3-digit format like
    "001", "015")
2. Explanation for each selected image (if there is any)

Output JSON sample format:
{{
   "selected_images": ["001", "005", "012"],
   "explanations": ["Explanation for image 001", "Explanation for image 005", "
       Explanation for image 012"] # Set this field to null if there is no explanation
}}

JSON Output:"
```

11.3. Intent Rating (visual context)

```
"You are a helpful assistant that rates images from an album.
Rating Query: {query}

Your task:
Rate ALL images based on how well they match the rating query above. Use a scale from
    0 to 3.

Rating Scale:
```

```
0 = Not relevant - Image does not match the query at all
1 = Somewhat relevant - Image has minor or tangential relevance to the query
2 = Relevant - Image clearly matches the query
3 = Highly relevant - Image is an excellent or perfect match for the query

Output format - you MUST follow this exactly:
Image-<id>: <rating>

Requirements:
Rating Criteria:
- Rate ALL images provided - do not skip any image
- Use only integers: 0, 1, 2, or 3 (no decimals or ranges)
- Base ratings strictly on the query criterion, not general image quality
- Be consistent in your rating standards across all images
- Consider the primary subject/content of each image when rating

Distribution Guidelines:
- Reserve rating 3 for images that strongly exemplify the query
- Use rating 0 for images clearly unrelated to the query
- Try not to rate all images the same - differentiate when possible

Format Requirements:
- Each line: Image-<number>: <rating>
- One image per line, no extra text or explanations
- Maintain numerical order of image IDs
- No extra commentary outside the format

Now give your ratings:"
```

Post-processing:

```
"Task: Parse the following and convert it to structured JSON format: <model_output>
Extract these exact info:
1. Image IDs and their corresponding ratings (convert image ID to 3-digit format like
    "001", "015")
2. Explanation for each rating (if there is any)

Output JSON sample format:
{{
   "images": ["001", "002", "003", "004", "005"],
   "ratings": [3, 2, 1, 0, 3],
   "explanations": ["Explanation for the rating of image 001", "Explanation for the
       rating of image 002", "Explanation for the rating of image 003", "Explanation
       for the rating of image 004", "Explanation for the rating of image 005"] # Set
       this field to null if there is no explanation
}}

JSON Output:"
```

11.4. Intent Rating (language context)

```
"You are a helpful assistant that rates images from an album.
Rating Query: {query}

Your task:
Rate this specific image based on how well it matches the rating query above. Use a
    scale from 0 to 3.
```

```
Rating Scale:
0 = Not relevant - Image does not match the query at all
1 = Somewhat relevant - Image has minor or tangential relevance to the query
2 = Relevant - Image clearly matches the query
3 = Highly relevant - Image is an excellent or perfect match for the query

Output format - you MUST follow this exactly:
Answer: <rating>

Requirements:
Rating Criteria:
- Use only integers: 0, 1, 2, or 3 (no decimals or ranges)
- Base rating strictly on the query criterion, not general image quality
- Consider the primary subject/content of the image when rating

Format Requirements:
- Single line format: Answer: <rating>
- No additional text or explanations
- Rating must be 0, 1, 2, or 3

Now give your rating:"
```

Post-processing:

```
"Task: Parse the following and convert it to structured JSON format: <model_output>
The input contains multiple lines, each representing one image's rating in format "
    Image-X: Answer: <rating> - explanation" or "Image-X: Answer: <rating>".

Extract these exact info:
1. Image IDs and their corresponding ratings (convert image ID to 3-digit format like
    "001", "015")
2. Explanation for each rating (if there is any)

Output JSON sample format:
{{
   "images": ["001", "002", "003", "004", "005"],
   "ratings": [3, 2, 1, 0, 3],
   "explanations": ["Explanation for the rating of image 001", "Explanation for the
       rating of image 002", "Explanation for the rating of image 003", "Explanation
       for the rating of image 004", "Explanation for the rating of image 005"] # Set
       this field to null if there is no explanation
}}

JSON Output:"
```

11.5. Group Labeling (visual context)

```
"You are a helpful assistant that groups images from an album.
Grouping Query: {query}
Available categories: {categories}

Your task:
1. Assign each image to exactly ONE category from above
2. If there is a 'misc' or 'miscellaneous' category, put all images that don't fit any
     other category here

Output format - you MUST follow this exactly:
Group-<number>: <category name>
```

```
[Image-X, Image-Y, Image-Z]

Example (if available categories are: ["A", "B", "C", "Misc"]):
Group-1: A
[Image-1, Image-4, Image-7]
Group-2: B
[Image-2, Image-5]
Group-3: C
[Image-8, Image-9]
Group-4: Misc
[Image-3, Image-6]

Requirements:
- Use ONLY categories from the provided list - {categories}
- Each category must contain at least ONE image
- Each image must be assigned to exactly ONE category

Format Requirements:
- Output groups in the SAME ORDER as given in the list above
- List ALL image IDs for each group in square brackets with commas
- Maintain the sequential group numbering (1, 2, 3, ...)
- Maintain the sequential image numbering within each group (1, 2, 3, ...)
- No extra text, explanations, or commentary

Now group the provided images:"
```

Post-processing:

```
"Task: Parse the following and convert it to structured JSON format: <model_output>
Extract these exact info:
1. Groups of image IDs (each group as a list, convert image IDs to 3-digit format like
     "001", "015")
2. Category for each group

Output JSON sample format:
{{
   "groups": [
       {{
           "images": ["001", "010", "015"],
           "category": "Ceremony photos"
       }},
       {{
           "images": ["008", "009", "012"],
           "category": "Reception photos"
       }},
       {{
           "images": ["003", "006", "007"],
           "category": "misc"
       }}
   ],
   "total_groups": 3
}}

JSON Output:"
```

11.6. Group Labeling (language context)

```
"You are a helpful assistant that assigns images to groups.
```

```
Grouping Query: {query}
Available categories: {categories}

Your task:
Determine which ONE category this specific image belongs to from the available
    categories above.

Output format - you MUST follow this exactly:
Answer: <category-name> - <brief explanation>

Requirements:
- Select ONLY ONE category from the provided list
- Category name must exactly match one from the available categories
- If the image doesn't fit any specific category well, assign it to 'Misc' or '
    Miscellaneous' if available
- Provide a brief explanation (10-20 words) justifying your choice
- Be specific about what features or content make this image fit the chosen category
- Keep explanation concise and factual

Format Requirements:
- Single line format: Answer: <category-name> - <brief explanation>
- Category name must match exactly (case-sensitive)
- No additional text before or after
- Must choose from: {categories}

Now assign this image to a category:"
```

Post-processing:

```
"Task: Parse the following and convert it to structured JSON format: <model_output>
The input contains multiple lines, each representing one image's group assignment in
    format "Image-X: Answer: <category-name> - explanation" or "Image-X: Answer: <
    category-name>".

Extract these exact info:
1. Groups of image IDs organized by their assigned categories (convert image IDs to 3-
    digit format like "001", "015")
2. Category for each group

Output JSON sample format:
{{
   "groups": [
       {{
           "images": ["001", "010", "015"],
           "category": "Unwrapping Presents"
       }},
       {{
           "images": ["008", "009", "012"],
           "category": "Birthday Cake"
       }},
       {{
           "images": ["003", "006", "007"],
           "category": "misc"
       }}
   ],
   "total_groups": 3
}}

JSON Output:"
```

11.7. Group Clustering (visual context)

```
"You are a helpful assistant that groups images from an album.
Grouping Query: {query}

Your task:
1. Analyze all images and identify distinct categories based on the grouping query
    above
2. Output the list of categories
3. Assign each image to exactly ONE category that best matches the query
4. If images don't fit any main category, place them in a 'Misc' category

Output format - you MUST follow this exactly:
Categories: [category1, category2, category3, ...]
Group-<number>: <category name>
[Image-X, Image-Y, Image-Z]

Example:
Categories: [A, B, Misc]
Group-1: A
[Image-1, Image-4, Image-7]
Group-2: B
[Image-2, Image-5]
Group-3: Misc
[Image-3, Image-6]

Requirements:
- Generate less than 5 categories
- Each category must be unique and distinct with no conceptual overlap
- Categories should be mutually exclusive (e.g., don't create both "People" and "
    Portraits")
- Each category must contain at least ONE image
- Category names must be specific and descriptive, not vague (avoid names like "Other
    ", "Stuff", "Things", "Events")
- List ALL image IDs for each category in square brackets with commas
- Every image must appear in exactly one category - no image should be left out or
    duplicated
- Prioritize balanced grouping over creating many small groups
- The category names should directly relate to the grouping query


Now group the provided images:"
```

Post-processing:

```
"Task: Parse the following and convert it to structured JSON format: <model_output>
Extract these exact info:
1. List of all categories (if there's a line starting with "Categories:", extract the
    category names from it)
2. Groups of image IDs (each group as a list, convert image IDs to 3-digit format like
     "001", "015")
3. Category for each group

Output JSON sample format:
{{
   "categories": ["Ceremony photos", "Reception photos", "misc"],
```

```
   "groups": [
       {{
           "images": ["001", "010", "015"],
           "category": "Ceremony photos"
       }},
       {{
           "images": ["008", "009", "012"],
           "category": "Reception photos"
       }},
       {{
           "images": ["003", "006", "007"],
           "category": "misc"
       }}
   ],
   "total_groups": 3
}}

Note: Set "categories" field to null if there's no "Categories:" line in the input.

JSON Output:"
```

12. Limitations of CUFED

Our benchmark is built on albums sourced from CUFED, which consists of 23 event categories spanning important personal events and group activities, such as weddings, birthdays, casual family gatherings, beach trips, and sports. In our setting, an "album" is defined as a set of photos associated with a particular time-bounded life event, rather than a full personal gallery accumulated over long periods of time. As a result, the benchmark does not cover all forms of real-world photo collections, especially mixed galleries containing unrelated everyday photos, cross-event collections, or albums with highly heterogeneous curation styles. At the same time, we believe CUFED remains relevant for studying album-level understanding, since the life events it captures are still common today and modern photography has increasingly shifted toward mobile-centric practices that align well with this type of data. Finally, CUFED was preprocessed to remove near-duplicate images before annotation; we clarify that near-duplicate detection or removal is not part of the capabilities evaluated by AlbumBench, and performance on our tasks should not be interpreted as measuring that ability.

13. Ethical Statements

13.1. Dataset Source and Licensing

This research utilizes the CUFED dataset, an open-source dataset distributed under a Creative Commons license. The dataset was collected with appropriate consent from participants and has been widely used in the computer vision community. All images in our benchmark are derived from this publicly available dataset, and we maintain the same licensing terms. By using an established, ethically-sourced dataset, we ensure that proper consent protocols were followed during the original data collection process.

13.2. Bias Mitigation in Annotations

We recognize that annotation tasks involving subjective judgments, such as selecting or rating image based on a query, carry inherent risks of bias. To minimize subjectiveness and bias in our labeling process, we practice the following:

1. Quality control/review on annotations to reduce individual bias
2. Diverse annotator pool in terms of companies and backgrounds
3. Clear annotation guidelines that define objective criteria where possible
4. Instructions that emphasize task-specific objectives rather than personal preferences

We acknowledge that some subjectivity is inherent to album organization tasks, as user intent and preferences vary naturally. Our annotations aim to capture this diversity while maintaining consistency in how tasks are interpreted and executed.