Robusto-1 Dataset: Comparing Humans and VLMs on real out-of-distribution Autonomous Driving VQA from Peru

Supplementary Material

6.1. Human Protocol VQA

A total of nine humans participated in this small pilot experiment as *volunteers*. A consent digital consent form was given to the volunteers where they were briefly told about the goals of the study. Participants were required to perform the task on a computer or laptop and were not allowed to use their phones to ensure a wider field of view, as watching a video on a phone may result in missing key elements in such short clips. Responses were recorded digitally and stored anonymously with encrypted participant IDs. Participants provided their digital consent by ticking on a box in a Google Forms spreadsheet to share their data in anonymized way for research and commercial purposes.

The participant demographics consisted of nine individuals aged 18 to 35 from Peruvians living in Peru. Subjects were recruited as a mixture of friends and colleagues of the authors through open advertising in different group chats. The participants had varying levels of driving experience and were fluent in English, as the questions were asked and answered in English. Participants also digitally confirmed their english fluency, and participants who did not have such were potentially going to be removed from the analysis. This was not the case and we analyzed all 9 subjects in the experiment. It is important to note that the VLMs were also tested using the same questions in English. All participants were Peruvian. We are aware that the small study group is interesting because Peruvians generally speak spanish (not english), and that VLMs have likely not seen dashcam driving data in Peru (a spanish speaking country). A future study will include English-speaking participants (*e.g.* Americans) and show them a mixture of data from both people driving in the United States and Peru to study the interaction of language fluency and dashcam data provenance to the study.

Interestingly, some participants at the end of the experiment thought it was a text-base labelling task (given what they have read in the news about manual bounding-box labeling being required to train AI models), as they were unfamiliar with Question-Answering (QA) research. Approximately half of the participants reported via email, Slack or WhatsApp that many questions seemed subjective – however, this is not a negative comment, as it verifies the intention of our experiment to push the boundary of human interpretability through OOD stimuli with questions of varying level of subjectiveness such as the hypotheticals & counterfactuals in Block 3.

6.2. Multimodal Input Processing Pipeline

To systematically evaluate the ability of Vision-Language Models (VLMs) to analyze driving scenes, we implemented a input processing protocol tailored to the specific requirements of each model. The objective was to ensure that all models received equivalent multimodal input while respecting their individual API constraints and format requirements. This protocol enabled us to compare their performance fairly across tasks involving video-based visual question answering (VQA).

Each model was provided with a series of frames extracted from driving videos alongside a set of structured questions. Given the variability in how different VLMs process visual inputs, we employed a prompt adaptation mechanism that converts video data into a compatible format for each model.

Below, we describe the input processing strategy for each VLM tested in our experiments.

CogVLM. For CogVLM, video data were submitted as complete binary files. The input video was read from a local file in binary mode and passed along with an adapted prompt via the Replicate API. The input dictionary included keys for the prompt, the binary video file (“input_video”), and generation parameters (“top_p” set to 0.9, “temperature” set to 1, and “max_new_tokens” of 2000).

Qwen 2. For Qwen2, videos were hosted remotely. Each video was downloaded using HTTP requests, converted into an in-memory file using Python’s `BytesIO` module, and then combined with the adapted prompt to form the input. These data were sent to the model via the Replicate API in a similar structure as for CogVLM, enabling Qwen2 to process video inputs directly from remote sources.

Pixtral. Pixtral Large model processes video content by analyzing individual frames rather than receiving a complete video file as a single input. For each video, frames were extracted at a rate of 1 FPS and converted to Base64-encoded JPEG strings. The resulting input was constructed as a message comprising a text component (the adapted prompt) followed by a series of image components. Each image was represented by a Base64 string prefixed with `""`.

DeepSeekV3. DeepSeek V3 was evaluated by extracting video frames at 10 FPS and converting each frame into a Base64-encoded string. The adapted prompt was combined with a list of these image strings (each prefixed with `""`) into a message structure, which was then submitted to the model via its API.

Gemini. Gemini processes video inputs by first combining the system prompt with a marker that denotes the start of the visual sequence. Each video frame is then converted into an image component via the API's `Part.from_image` method, and a text component containing the user prompt is appended. Gemini 2.0 was deployed on Google Cloud Platform (GCP) through Vertex AI, utilizing the checkpoint "gemini-2.0-flash-exp" in accordance with the guidelines provided in the Vertex AI Generative Models documentation. The LLM generation parameters were set to a maximum of 100 tokens, a temperature of 1.0, and a top-p of 0.9.

Llama. For Llama-based models, our protocol transforms each video frame into a Base64-encoded JPEG string that is then integrated with the system instructions and user prompt into a single text block. This combined input is submitted to the model via its API. In our experiments, Llama 3.2 was deployed in GCP using Vertex AI, employing the checkpoint "Llama-3.2-11B-Vision-Instruct-meta." The LLM generation parameters were set to a maximum of 100 tokens, a temperature of 1.0, and a top-p of 0.9.

6.3. MetaData & Tags

The distribution of driving scenarios suggested that we create a pre-fixed list of 16 meta-tags from which we manually annotate certain properties from a video clip. Sample meta-data attributes are: *1. Vehicle Action*, *2. Driving Action Reasoning*, *3. Vehicle Motion Behavior*, *4. Traffic Signs*, *5.Traffic Lights*, *6. Weather Conditions*, *7. Road Surface Conditions*, *8. Road Structures*, *9. Static Objects*, *10. Other Vehicle Behaviors*, *11. Pedestrian Behaviour*, *12. Unexpected Obstacles*, *13. Emergency Situations*, *14. Lighting Conditions*, *15. Traffic Conditions*, *16. Driving Environment*. The full list of information of the labels derived from the meta-data attributes can be seen in the Table 4. These meta-tags are available for all 200 videos, and the 7 external ones used in the study of our paper, and were used as the basis for prompting the Oracle LLM the variable questions.

6.4. Question Generation Details with LLMs

To assess the current gap in the ability of Language Models to understand driving scenes, we designed a process for generating context-specific questions for each video. This process focuses on the first block of queries, termed "Variable", one of three blocks used in our experiments (the other two being "Multiple Choice" and "Counterfactual & Hypothetical"). In this block, each video is associated with a set of five targeted questions, along with concise answers, that are derived solely from the metadata manually curated for the corresponding driving scene.

Initially, we compiled a database containing key metadata for each video. This metadata includes general information such as the sample identifier, scene location, ego vehicle details (e.g., vehicle actions, motion behavior), and external factors (e.g., traffic signs, weather conditions, road surface conditions).

For each video, the curated metadata is stored in a JSON file that is subsequently processed using GPT-based models accessed through the ChatGPT platform (specifically, through https://chatgpt.com/gpts). Our approach leverages customizable GPTs, which are configured through two primary components: detailed system instructions and an initial conversation starter phrase. The system instructions explicitly guide the model to generate five relevant questions based exclusively on the provided metadata, while the starter phrase establishes the context for the conversation, ensuring consistency and clarity throughout the exchange.

The instructions provided to the GPT are as follows:

```
You are an AI assistant specialized in analyzing
driving scenarios. You will receive a list of
JSON objects, each containing partial metadata
about different driving scenes. Be aware that
the provided data is incomplete, and important
elements of the scenes may be missing.

For each JSON sample, your task is to:
1. Read the JSON object.
2. Include the "#" and "Name" from the JSON
object at the beginning to indicate which sample
you are analyzing.
3. Generate **five** relevant and contextually
appropriate questions based solely on the
information available in the JSON object.
4. Provide short and direct answers to each
question.

Focus on what is observed in the scene according
to the metadata, and consider that there might
be elements not explicitly mentioned.

Example format:

Sample #: 1
Name: 2023_01_10_153834_044_clip_00_16_100

Q1: [Question 1]
A1: [Answer 1]

Q2: [Question 2]
A2: [Answer 2]

Q3: [Question 3]
```

```
A3: [Answer 3]

Q4: [Question 4]
A4: [Answer 4]

Q5: [Question 5]
A5: [Answer 5]
```

The conversation begins with the following starter prompt, which underscores the need to analyze each JSON sample individually:

```
Below is a list of JSON samples, each containing
partial information about different driving
scenes. Please analyze each sample individually.
For each one:

- Generate five relevant questions based on the
metadata.
- Provide short and direct answers to each
question.

Remember that the metadata may be incomplete,
and consider the possibility that there are
other elements not mentioned in the file.
[Insert the list of JSON samples here]
```

6.5. Testing Frame Processing Capacity

We conducted a synthetic experiment to evaluate whether each LLM could correctly interpret the temporal sequence of frames and detect objects introduced at specific moments. A series of frames was generated depicting a red ball on a white background moving diagonally from the bottom-left to the top-right corner. The objective was to verify whether the models could infer the ball's direction by processing the frames in the correct temporal order.

Additionally, we introduced a green star in one frame at a time to assess whether the models were capable of examining all frames throughout the sequence. In each iteration of the experiment, the green star was inserted into a different frame. If a model accurately recognized the presence of the green star, it suggested that the model had successfully processed that particular frame rather than skipping or averaging across the sequence.

The questions posed to the models focused on identifying the direction of the movement of the red ball and specifying if other objects were present in the frames. The following prompt was used in each iteration:

```
Task: Answer the following questions based
solely on the sequence of images provided. The
images represent frames from a short video
sequence.

Questions:
1. In which direction is the red ball moving?
2. Do you see any other objects besides the red
ball? If so, please describe the object(s) and
their color(s).

Instructions:
- Carefully analyze each image frame by frame.
- Base your answers only on what is visibly
present in the images.
- Do not assume any information that is not
directly observable.
- Provide a concise answer, and explain your
reasoning if necessary.
```

By repeating this process for multiple iterations (placing the green star in different frames each time) and examining the models' responses, we assessed whether they could track the trajectory of the red ball and the newly introduced object without overlooking any part of the video.

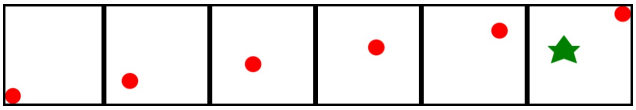

Figure 7. Images used to analyze the model's temporal understanding.

6.5.1 Results

The results confirmed that Pixtral supports a maximum of six frames per input, which means it did not successfully process the test at a frame rate of 10 fps. However, when tested at 1 fps, it demonstrated accurate frame sequence recognition, including the detection of the green star in the final frame.

On the other hand, Deepseek was tested at 10 fps and exhibited performance comparable to other models in terms of general response. However, a key limitation was identified: Deepseek only supports OCR (Optical Character Recognition), meaning its analysis is restricted solely to textual content present in the images. Since the model does not process visual information beyond text, we infer that its performance was influenced by the filenames and image descriptions, which contained hints about the video content. In fact, when the file names were changed, the model completely lost its accuracy in responses, confirming that its performance relied on external textual information rather than a genuine understanding of the visual content. We highlighted this limitation in the results (Table 1), where Deepseek appears with a dagger symbol (†), indicating that while it accepts image inputs, it only processes them for OCR purposes rather than for *"true"* visual scene understanding.

Additionally, we evaluated Qwen2 and CogVLM using the Replicate platform, setting the frame rate to 10 fps. According to available benchmarks, these models can process longer videos at higher frame rates. However, we standardized the input to 10 fps to ensure a consistent comparison across models, providing each Vision-Language Model (VLM) with an equivalent amount of temporal information. While both models successfully passed the test, there is

evidence of internal processing mechanisms that influence how frames are interpreted. Due to this additional processing, these models are marked with an asterisk (*) in the results table to indicate potential differences in frame handling compared to other models.

Models	Name	Test Passed?		
		10fps	1fps	0.5fps
DeepSeek V3	"deepseek-chat"	✓ †	-	-
Pixtral	"pixtral-large-latest"	✗	✓	-
*Qwen	"Qwen2-VL-7B"	✓	-	-
*CogVLM	"cogvlm2-video"	✓	-	-
Gemini	"Gemini-2.0-flash-exp"	✓	-	-
Llama	"Llama-3.2-11B-Vision-Instruct"	-	-	✓

Table 1. Comparison of vision-language models, including test results. Models marked with * were run through the Replicate platform. Models marked with † have pseudo multi-modal capabilities (see Section 6.5).

6.6. Running Visual-Language Models

We conducted our experiments using six publicly available Vision-Language Models (VLMs): Deepseek, Pixtral, Qwen2, CogVLM, Gemini, and Llama. These models were developed by organizations from three different countries: the United States of America (Gemini and Llama), France (Pixtral), and China (DeepSeek, CogVLM, and Qwen). Below, we describe the key aspects of how each model was accessed, configured, and tested.

Qwen2 and CogVLM2. The Qwen2 [69] and CogVLM2 [24] models were accessed through the Replicate platform, which offers a straightforward interface for evaluating AI models. Despite their fee-based model usage, the cost per query proved minimal relative to other platforms and was justified given our limited set of video prompts.

Setting up and running the models was a straightforward process, as it did not require the installation of additional tools or the implementation of advanced configurations. However, the example Python script provided by Replicate per model was modified to enable its use through the API. The modifications were primarily aimed at ensuring that the input consisted of the trial dataset videos and the prompt which had already been processed as previously detailed. These queries were directly loaded into the system, allowing for the efficient generation of results in a near-instantaneous manner. In terms of performance, the response time for each model was approximately 9-16 seconds, ensuring a rapid turnaround for queries. Additionally, the estimated cost per query to CogVLM model was $0.000725 and to Qwen2 model was $0.000975 providing a reference for computational efficiency and resource allocation.

Both models demonstrated fast and consistent performance on basic visual and textual analysis tasks. However, certain limitations were observed when interpreting images repetitively, evidencing a low variability in their responses, since they responded exactly the same to the same image and text input. Despite this limitation, the accessibility and ease of use of Replicate was a valuable tool to run and test models without requiring significant computational resources.

Pixtral. We evaluated the "Pixtral Large" model using its official API, which offers complimentary and direct access to its functionalities. Following the official documentation, we integrated the Pixtral model through JSON-based requests to transmit images and prompts. On average, Pixtral required 1.5–2.8 seconds per query when the input consisted of five images plus a question. However, processing times increased for more complex images, such as those containing multiple overlapping objects or environments with variable lighting. In these cases, response times extended due to challenges in classifying secondary or out-of-distribution (OOD) objects.

In one specific test case, involving a counterfactual & hypothetical question and an urban scene with traffic and various unidentified objects on the street, Pixtral required approximately 9 to 16 seconds to generate a response, likely due to the complexity in the image.

The experiment with the 7 videos ended with a 99 % success rate in executing requests without errors (only one error was obtained during the experiment). Overall, Pixtral showed strong performance on tasks such as generating textual descriptions and variability in its responses without going out of context. In conclusion, the Pixtral API proved to be robust, user-friendly, and highly effective, making it a valuable tool for the development and evaluation of Vision-Language tasks.

DeepSeek-V3. DeepSeek V3 [40] was evaluated through its official API to assess its capability in visual and textual analysis tasks. The integration was carried out through JSON-based requests, achieving an average response time of 0.9 seconds per query, highlighting its speed compared to other models tested. The experiment used a frame rate of 10 images per second. For each query, 10 repetitions were performed to ensure consistency of the results.

Regarding token management, DeepSeek models use tokens as basic units to process text and as a basis for billing. A token can represent a character, word, number, or symbol. Approximately, the cost per query for us was 1200-1500 tokens. A query consists of a processed message/prompt and a set of 50 images. The prompt contains approximately 913

characters, and the images are in HD, with a resolution of 1920×1080 pixels. The exact number of tokens processed per query is determined based on the model's response.

A publicly available tokenizer facilitated offline estimation of token usage, allowing for more efficient planning of model queries. DeepSeek's source code is available in its official GitHub repository, further enabling transparency and reproducibility.

Gemini. Gemini 2.0 was deployed on Google Cloud Platform (GCP) via Vertex AI, utilizing the checkpoint "gemini-2.0-flash-exp" to ensure seamless integration into our experimental pipeline. Our implementation followed the guidelines provided in the Vertex AI Generative Models documentation available at https://cloud.google.com/vertex-ai/generative-ai/docs/reference/python/latest/vertexai.generative_models. We tested this model with videos recorded at 1920×1080 resolution and 10 frames per second, encoding each frame prior to submission through the Vertex AI API. For each question on every video, the experiment was repeated 20 times to capture the variability in the LLM responses.

Llama. Llama 3.2 was deployed on Google Cloud Platform (GCP) via Vertex AI following the recommended guidelines for uploading pre-built models to the Model Registry and deploying them to a Vertex AI Endpoint. In our experiments, we used the checkpoint "Llama-3.2-11B-Vision-Instruct-meta." The model was deployed on an `a2-highgpu-1g` machine equipped with one NVIDIA Tesla A100 GPU. Video frames, provided in JPEG format, were used as inputs. Notably, this model exhibited a limitation in its processing capacity, as it was able to process only up to three frames per video. To capture the variability in the responses, each question for every video was repeated 20 times.

6.7. Sentence Embedding

To represent textual data in a high-dimensional vector space, we used a sentence embedding model that encoded semantic information while preserving contextual dependencies. The primary sentence embedding used for the plots presented in the main body of this paper was `all-mpnet-base-v2`, a transformer-based architecture pre-trained on large-scale corpora and optimized for semantic similarity tasks available in https://huggingface.co/sentence-transformers/all-mpnet-base-v2. To generalize our results, we re-ran our analysis using two other sentence embeddings such as `paraphrase-mpnet-base-v2` and `e5-large-v2` to illustrate the effects of different embeddings on the final pattern of results. These results for RSA can be seen in Figure 9. Both of these sentence embeddings are available in https://huggingface.co/sentence-transformers/paraphrase-mpnet-base-v2 and https://huggingface.co/intfloat/e5-large-v2 respectively.

6.8. Data Curation and Additional Analysis

There were certain cases for the multiple choice questions where the VLMs did not correctly answer one of the main responses, or answered with a small variant. For example, in some cases there are answers that only had Yes/No, that were responded with similar but no exact answers like "`Option: 'Yes'`", "`Option: ['No']`", "`Answer: Option: No`" or "`[No]`", etc. These variants of Yes/No were cured to be the same as Yes or No respectively.

For other multiple-choice questions, there were examples such as those for the clutter rating where the VLM responded to some false interval that was not in the options. For example, "`Option: 2 to 4`", "`Option: 1 to 5`", "`Option: More than 10`", "`Option: 10 or more`" or just "`Option: 9`". To curate the data, the solution was to review and contrast the original intervals we proposed as multiple-choice responses (Table 3) and verify whether the answers fit within the provided ranges. For example, "`Option: 1 to 5`", "`Option: More than 10`", or "`Option: 2 to 4`" did not fit into any of the established ranges. In such cases, the response was discarded and not considered for analysis. On the other hand, there were cases where the response *did* fit within one of the ranges, such as "`Option: 9`" or "`Option: 11-15.`" In this data curation process, we were strict in ensuring that the responses matched correctly.

As final results, we find a total of `1734/5460 (31.75%)` modifications in all Vision-Language Models (VLMs). On the other hand, responses that could not be included in the analysis were ignored and discarded. Ignored responses include, for example, those that did not fit within any of the predefined multiple-choice ranges. There were a total of `79/5460 (1.44%)` of ignored responses. Next, we will provide a detailed breakdown of the modifications and ignored responses for each VLM. Processed Data:

```
Llama-3.2 - Modifications: 350, Ignored: 2,
Total responses: 1050
cogvlm2 - Modifications: 22, Ignored: 0, Total
responses: 105
deepseek_v2 - Modifications: 327, Ignored: 44,
Total responses: 1050
gemini-2.0 - Modifications: 667, Ignored: 33,
Total responses: 2100
pixtral- Modifications: 350, Ignored: 0, Total
responses: 1050
qwen2 - Modifications: 18, Ignored: 0, Total
responses: 105
```

All results in the main body of this paper were done with the curated responses. However, we also re-did our analysis with the uncurated (raw) responses, and also using a single answer instead of the average (pooled) answer per query per VLM. Indeed, as can be seen in our raw data repository: Robusto-1, there are cases where some VLMs produce highly varying responses to the same questions. To address this variablilty (given that the embedding of several "Yes's and No's" can be "Maybe", and similarly for open response questions, we also re-did our analysis with a single responses, and found no large variation to the same pattern of results as using the pooled answer per VLM. We have added these main results in the supplementary plots.

Models	Name	API Access	Input Modality	Frame Rate (fps)
DeepSeek V3	deepseek-chat	Direct	Images & Text	10
Pixtral	pixtral-large-latest	Direct	Images & Text	1
Qwen2	Qwen2-VL-7B	Replicate	Video & Text	10
CogVLM	cogvlm2-video	Replicate	Video & Text	10
Gemini	Gemini-2.0-flash-exp	Direct	Images & Text	10
Llama	Llama-3.2-11B-Vision-Instruct	Vertex AI	Images & Text	0.5

Table 2. Summary of parameters and input modalities for evaluated Vision-Language Models. "API Access" indicates the method through which each model is accessed: Direct access via a dedicated API, or indirectly via external platforms such as Replicate or a custom deployment on Vertex AI.

Questions	
Question 1	Open-ended text response
Question 2	Open-ended text response
Question 3	Open-ended text response
Question 4	Open-ended text response
Question 5	Open-ended text response
Q6: Please rate the level of clutter from 1 to 10. Consider 10 as the highest level of clutter and 1 as the lowest.	1-10
Q7: Is this a recurrent driving scenario for you?	yes/no
Q8: Estimate how many pedestrians are there in the scene?	0,1, 2-3,4-6,7-10,11-20, 21+
Q9: Is this situation hazardous for the driver?	yes/no
Q10: On a scale of 1-10, how well do you think an autonomous vehicle would drive in this scene? Consider 10 as perfect driving and 1 as terrible driving.	1-10
Q11: What would have had to happen in this video for a crash to have occured involving the driver?	Open-ended text response
Q12: What would have had to happen in this video for an external crash to have occured not involving the driver?	Open-ended text response
Q13: Imagine if you had taken the opposite action in this scene (for example, braking instead of accelerating, or accelerating instead of braking). What do you think would have happened?	Open-ended text response
Q14: What would be the next action to perform a U-turn in the next frames if the driver was driving an ambulance instead?	Open-ended text response
Q15: What would be the next action to perform a U-turn in the next frames if the driver was driving a motorcycle instead?	Open-ended text response

Table 3. Overview of the questions and expected response formats, grouped into three categories: Variable (Questions 1–5), Multiple Choice (Questions 6–10), and Counterfactual & Hypothetical (Questions 11–15), as administered to human participants and Vision-Language Models.

Ego Vehicle		
Vehicle Actions	Single-Label	Describes the physical actions performed by the vehicle, such as turns, acceleration, braking, lane changes, etc. Its purpose is to capture the observable behavior of the vehicle in the scene.
Driving Action Reasoning	Multi-Label & Open-Ended	Explains the reasoning behind the vehicle's actions (e.g., stopping due to a pedestrian or changing lanes to avoid an obstacle). Its purpose is to provide the necessary context to understand why the observed actions were taken.
Vehicle Motion Behavior	Multi-Label	Describes the observable motion of the vehicle, such as steady driving, acceleration, or braking, based on the visual cues in the segment. Its purpose is to capture how the vehicle moves during the segment in a qualitative way, without requiring precise numerical values.
External Factors		
Traffic Signs	Multi-Label	Identifies and categorizes the traffic signs visible in the scene (e.g., stop signs, yield signs, speed limits). Its purpose is to evaluate how traffic signs influence the decisions of the driver and the vehicle.
Traffic Lights	Single-Label	Captures the state of the traffic light in the scene (red, green, yellow, off). Its purpose is to determine how the traffic light signals influence the vehicle's behavior.
Weather Conditions	Multi-Label	Describes the weather conditions during the driving event (e.g., fog, rain, sunny). Its purpose is to evaluate how weather conditions affect driving decisions and visibility.
Road Surface Conditions	Multi-Label	Describes the physical condition of the road, including potholes, poor maintenance, slippery surfaces, and temporary roadworks or debris. Its purpose is to evaluate how the road surface affects vehicle control and driving safety.
Road Structures	Multi-Label	Describes the physical infrastructure elements present on or alongside the road, such as islands, tunnels, and pedestrian crossings. Its purpose is to capture how these structures influence the driving behavior of the vehicle.
Static objects	Multi-Label & Open-Ended	Identifies buildings, poles, trees, and other static objects in the environment. Its purpose is to describe the urban or rural context surrounding the road.
Other Vehicle Behaviors	Multi-Label	Describes the interactions and maneuvers of external vehicles, including public transport, taxis, motorbikes, and private vehicles, and how they affect the driving decisions of the ego vehicle (e.g., lane invasion, sudden stops, overtaking). Its purpose is to capture the influence of other vehicles on the behavior of the ego vehicle.
Pedestrian Behavior	Multi-Label	Observes the behavior of pedestrians in the scene (crossing, waiting on the sidewalk, walking on the road). Its purpose is to capture how pedestrians interact with the vehicle and how they influence driving decisions.
Unexpected Obstacles	Multi-Label & Open-Ended	Describes any unexpected object or situation on the road, such as improperly parked vehicles, street vendors, or animals. Its purpose is to identify uncommon events that may affect driving.
Emergency Situations	Single-Label	Describes emergency situations or rare events that require a rapid response (accidents, roadblocks, roadworks). Its purpose is to identify incidents that alter the normal flow of traffic and require immediate attention.
Lighting Conditions	Single-Label	Describes the lighting conditions in the scene, such as natural lighting, street lighting, poorly lit areas. Its purpose is to evaluate how visibility affects driving decisions.
Traffic Conditions	Single-Label	Describes the state of traffic on the road, such as free-flowing, congested, stopped. Its purpose is to evaluate how traffic density affects driving decisions.
Driving Environment	Single-Label	Describes the environment in which the driving takes place, including areas that may affect vehicle behavior, such as school zones, markets, construction sites, or rural areas. Its purpose is to capture how the driving environment influences driving decisions.

Table 4. Driving scene attributes used as meta-data for LLM Q&A formulation. The table lists attributes grouped under Ego Vehicle and External Factors, indicating the label type (Single-Label or Multi-Label, with some requiring open-ended responses) and providing a description of each attribute's purpose in capturing different aspects of the driving scenario.

Figure 8. A collection of sample frames from 7 held-out videos used in our experiments form the Robusto-1 dataset. There is a combination of rural and urban scenes that humans and VLMs view. This preliminary study focused only on showing humans and machines 7 videos, but the dataset is composed of 200 additional videos (See Supplement) that we are releasing to the public for further research and experiments.

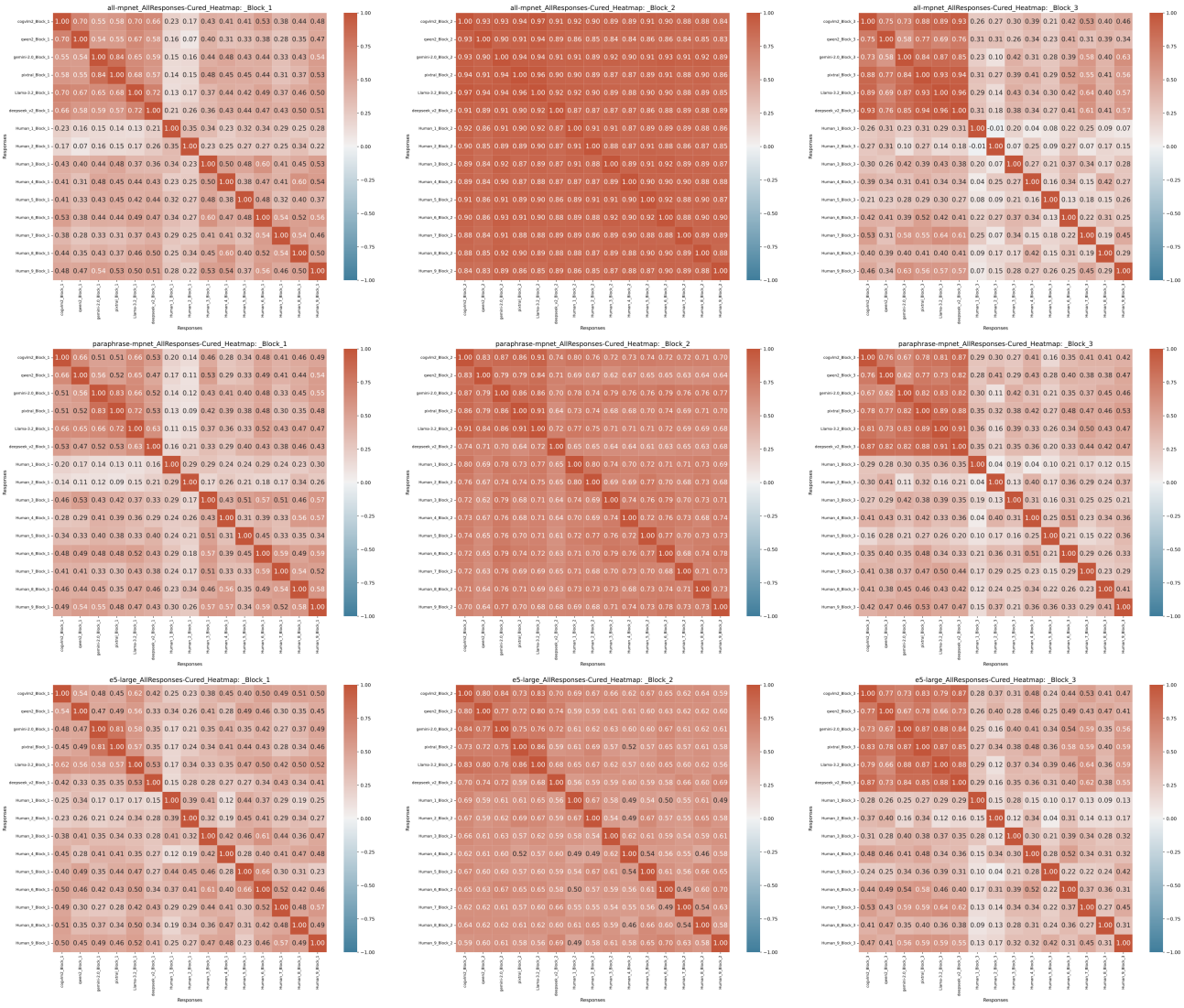

Figure 9. A collection of all the RSA plots for the 3 different types of embeddings used in the paper (all-mpnet, paraphrase-mpnet, e5-large). We observe that the pattern of results stays of our initial analysis stays the same with different levels of intensity.

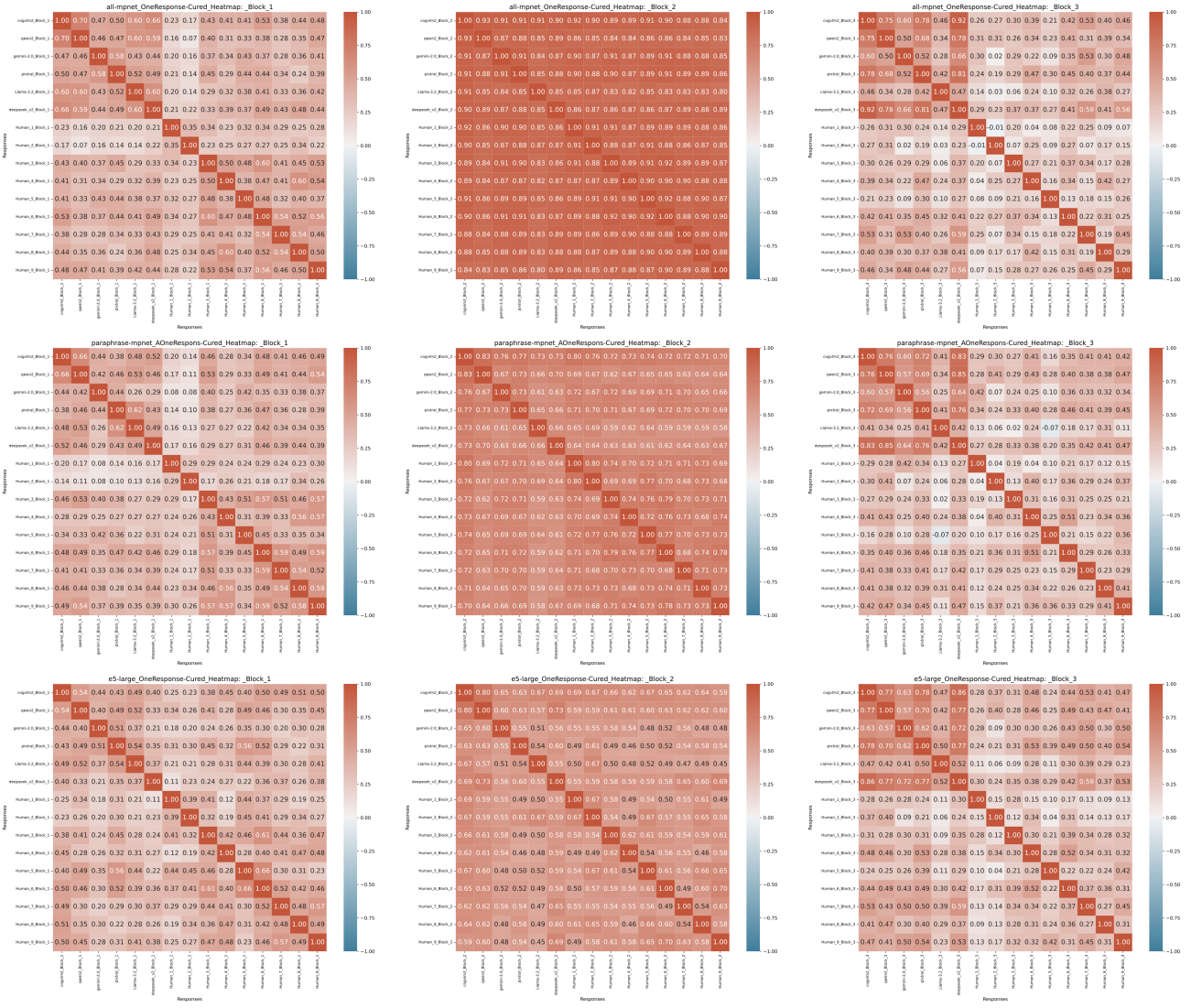

Figure 10. In this graph however we show how the RSA results would have looked like if we had just used one response rather than pooled (averaged) several observations per answer (also for all embeddings: all-mpnet, paraphrase-mpnet, e5-large). We find a very similar trend to the pooled responses for the VLMs. Though it would appear that pooling answers shows greater level of convergence across VLMs.

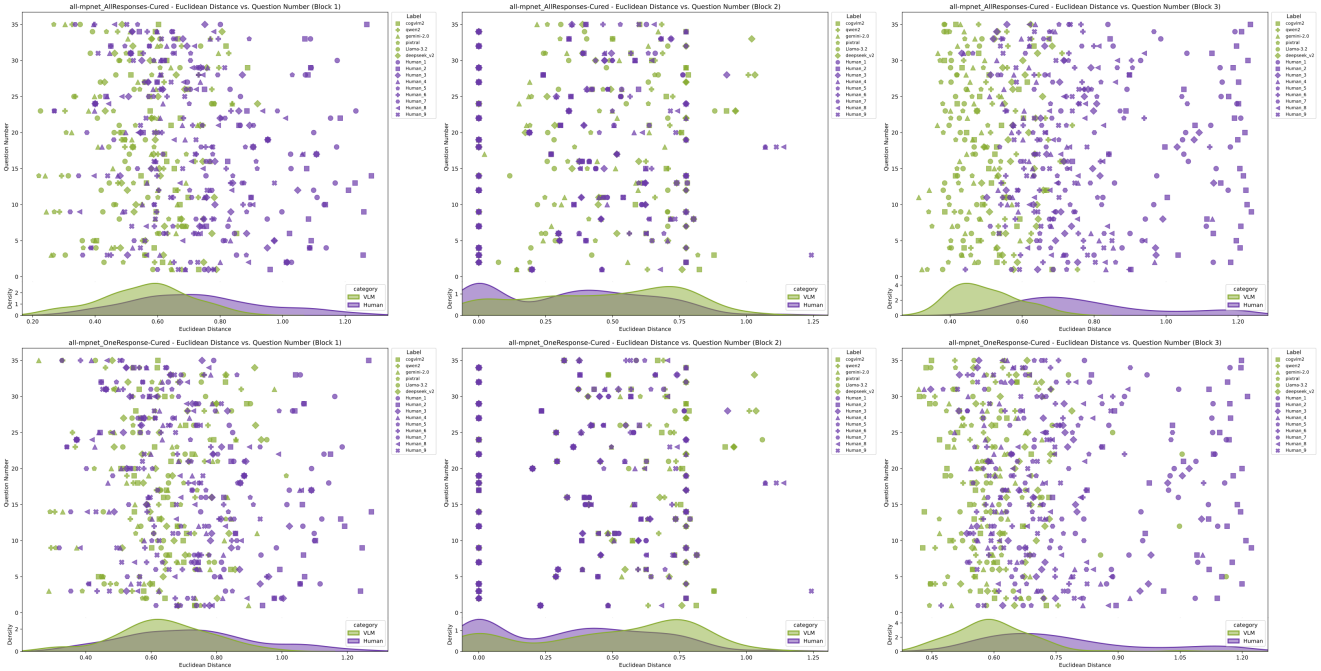

Figure 11. The Distance to the Median comparison of using a pooled vs single embedding is used across all systems (in particular the VLM). The same pattern of results holds for all-mpnet.

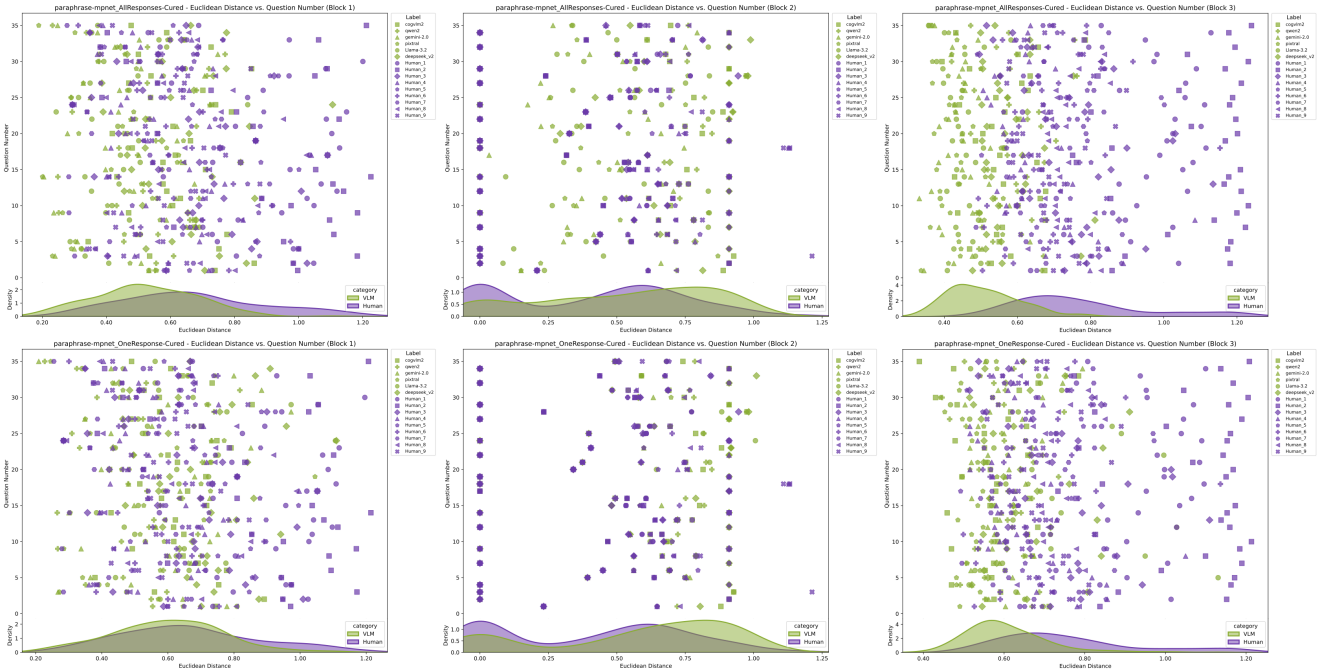

Figure 12. The Distance to the Median comparison of using a pooled vs single embedding is used across all systems (in particular the VLM). The same pattern of results holds for paraphrase-mpnet.

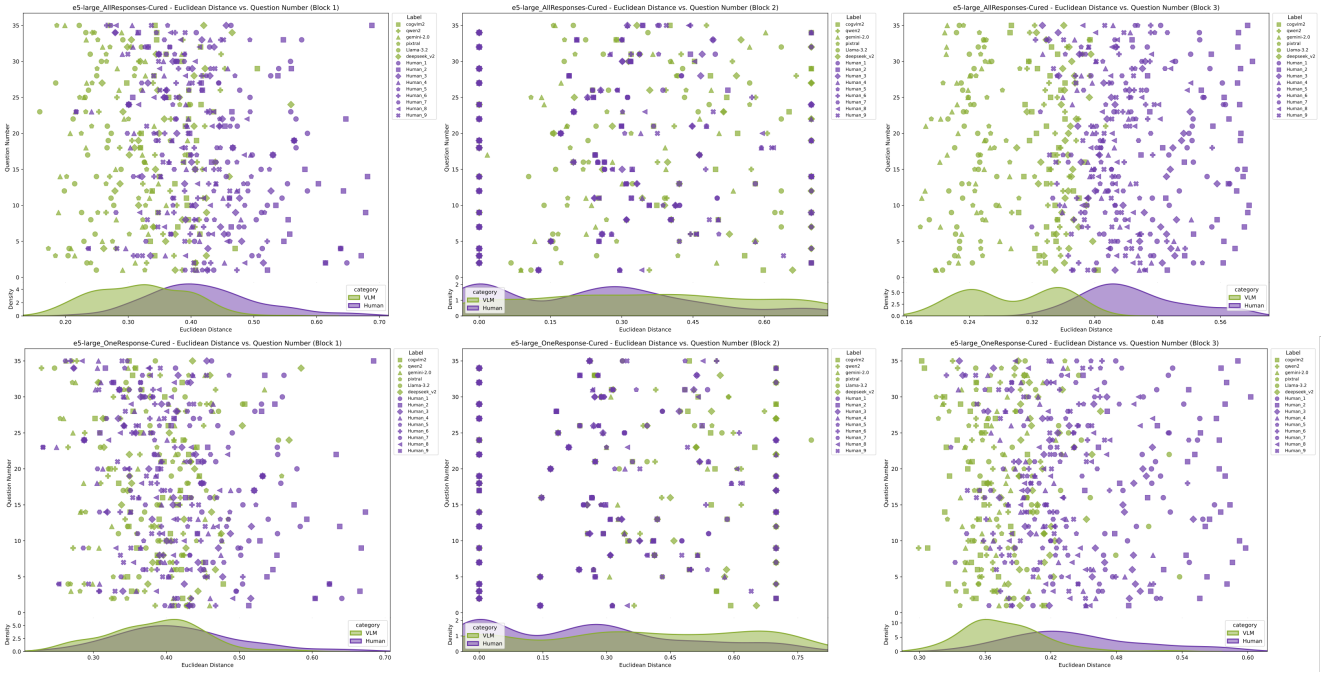

Figure 13. The Distance to the Median comparison of using a pooled vs single embedding is used across all systems (in particular the VLM). The same pattern of results holds for e5-net.

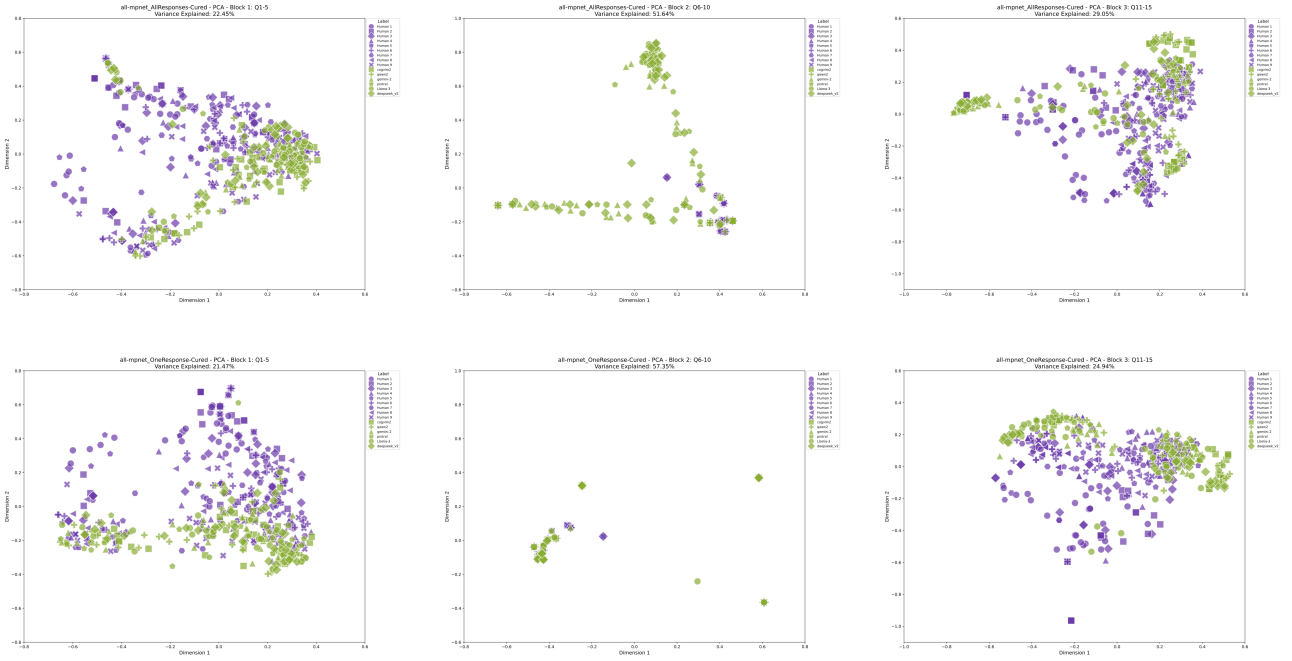

Figure 14. The PCA visualization of a comparison of using a pooled vs single embedding is used across all systems (in particular the VLM). The same pattern of results holds for all-mpnet.

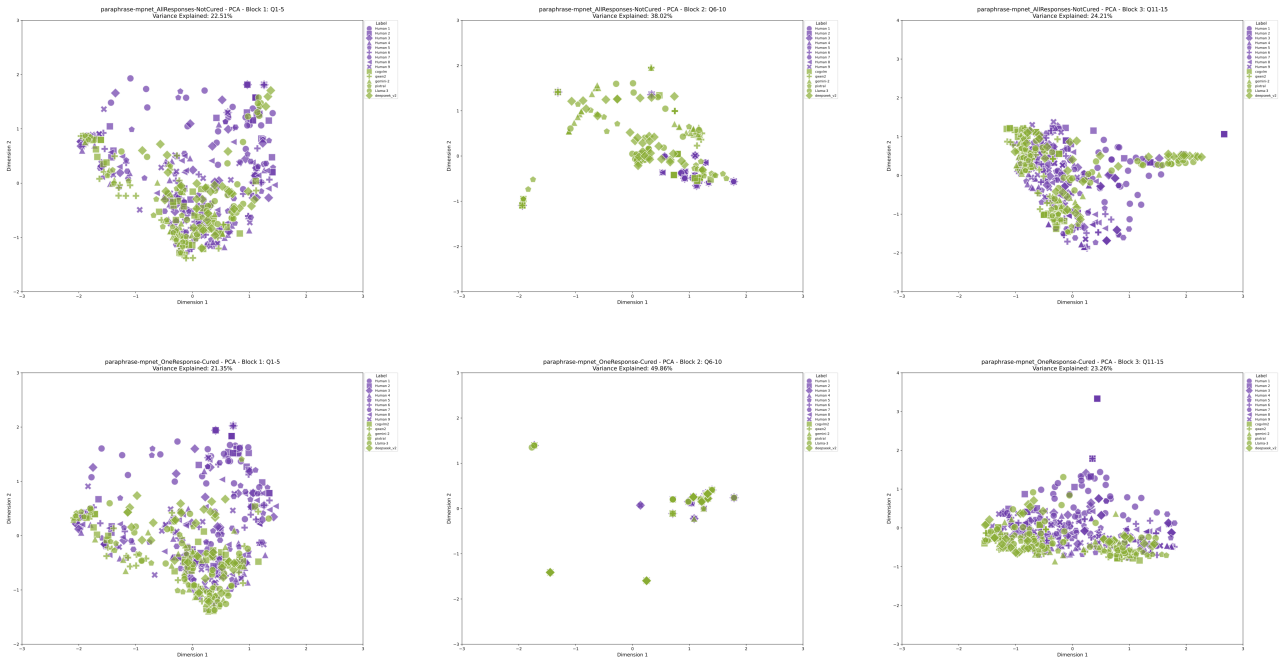

Figure 15. The PCA visualization of a comparison of using a pooled vs single embedding is used across all systems (in particular the VLM). The same pattern of results holds for paraphrase-mpnet.

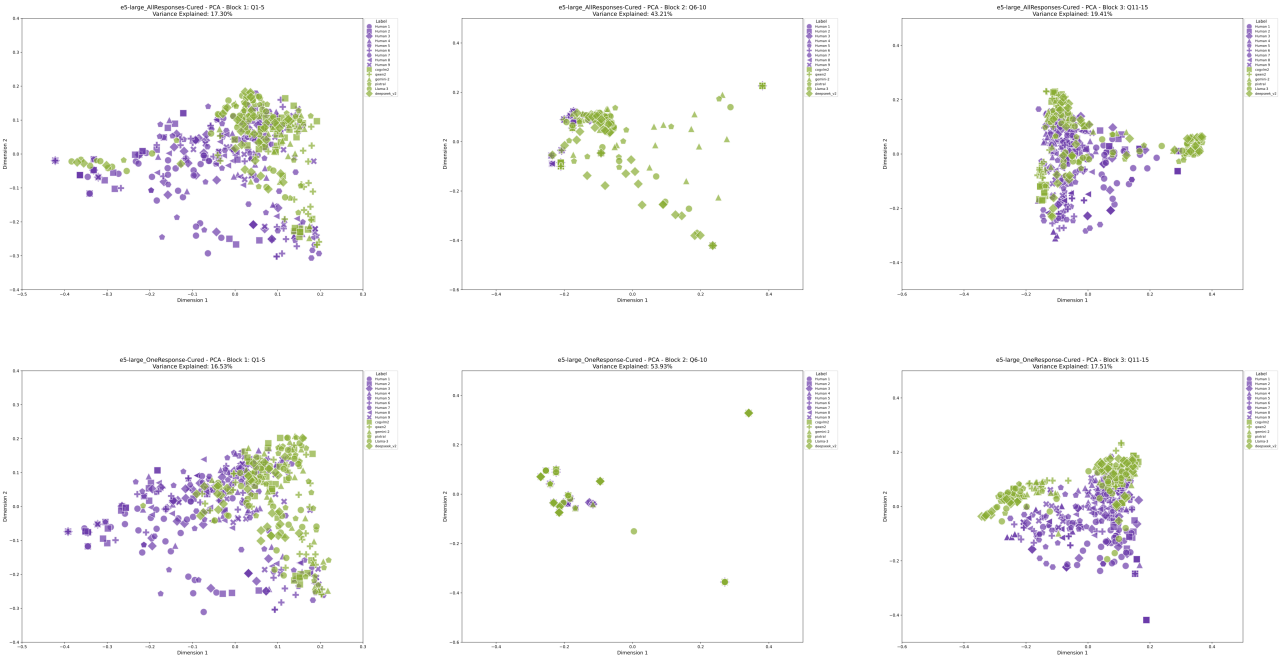

Figure 16. The PCA visualization of a comparison of using a pooled vs single embedding is used across all systems (in particular the VLM). The same pattern of results holds for e5-net.