Appendix

Supplementary Material

9. Temporal Logic Operation Example

Given a set of atomic propositions $\mathcal{P} = \{\text{Event A}, \text{Event B}\}$, the TL specification $\Phi = \square$ Event A (read as "Always Event A") means that 'Event A' is `True` for every step in the sequence. Additionally, $\Phi = \diamondsuit$ Event B (read as "eventually event b") indicates that there exists at least one 'Event B' in the sequence. Lastly, $\Phi =$ Event A U Event B (read as "Event A Until Event B") means that 'Event A' exists until 'Event B' becomes `True`, and then 'Event B' remains `True` for all future steps.

10. Prompt Understanding via Temporal Logic Specification (*PULS*)

In order to obtain $\mathfrak{D}_{\text{train}}$, we first begin with the larger dataset $\mathfrak{D}$ with size $\mathcal{B}$ where $\mathfrak{D} = \mathfrak{D}_{\text{T2P}} \cup \mathfrak{D}_{\text{T2TL}}$. $\mathfrak{D}_{\text{T2P}}$ and $\mathfrak{D}_{\text{T2TL}}$ are defined as the following:

$$\mathfrak{D}_{\text{T2P}} = \{(\mathcal{T}_i, \mathcal{P}_i)\}_{i=1}^{\mathcal{B}}, \quad \mathfrak{D}_{\text{T2P}} \subset \mathfrak{D}, \tag{11}$$

$$\mathfrak{D}_{\text{T2TL}} = \{(\mathcal{T}_i, \mathcal{P}_i, \Phi_i)\}_{i=1}^{\mathcal{B}}, \quad \mathfrak{D}_{\text{T2TL}} \subset \mathfrak{D}. \tag{12}$$

Using these datasets, we use *PULS* to find a specification Φ for each mode and for a given text prompt $\mathcal{T}$ using the following algorithm.

Algorithm 2: *PULS*

Require: LLM Prompt Optimizer `MIPROv2`
Input: List of mode M, text prompt $\mathcal{T}$, training examples $\mathfrak{D}_{\text{T2P}}$ and $\mathfrak{D}_{\text{T2TL}}$, number of few shot examples $\mathcal{N}$
Prompts: Modules LM_{T2P} and LM_{T2TL}
Output: TL Specification Φ

begin
  $\Phi \leftarrow \{\}$ `// Initialize empty set` Φ
  for $m \in M$ **do**
    $\mathfrak{D}_{\text{T2P|train}} = \texttt{MIPROv2.optimize}(\mathfrak{D}_{\text{T2P}}, m, \mathcal{N})$ `// Find optimal fewshot dataset`
    $\mathfrak{D}_{\text{T2TL|train}} = \texttt{MIPROv2.optimize}(\mathfrak{D}_{\text{T2TL}}, m, \mathcal{N})$ `// Find optimal fewshot dataset`
    $\theta^{\star}_{\text{T2P}} \leftarrow \{(\mathcal{T}_i, \mathcal{P}_i) \mid \mathcal{T}_i, \mathcal{P}_i \in \mathfrak{D}_{\text{T2P|train}}\}_{i=1}^{\mathcal{N}}$ `// Update parameters`
    $\theta^{\star}_{\text{T2TL}} \leftarrow \{(\mathcal{T}_i, \mathcal{P}_i, \Phi_i) \mid \mathcal{T}_i, \mathcal{P}_i, \Phi_i \in \mathfrak{D}_{\text{T2TL|train}}\}_{i=1}^{\mathcal{N}}$ `// Update parameters`
    $\mathcal{P} = LM_{\text{T2P}}(\mathcal{T}, m, \theta^{\star}_{\text{T2P}})$ `// Compute propositions`
    $\Phi \leftarrow \Phi \cup \{LM_{\text{T2TL}}(\mathcal{T}, \mathcal{P}, m, \theta^{\star}_{\text{T2TL}})\}$ `// Compute specification`
  end for
  return Φ

10.1. DSPy & MIPROv2

To evaluate L4-5 of Algorithm 2, *PULS* uses DSPy and MIPROv2 to optimize the prompts by selecting the appropriate subset of $\mathfrak{D}_{\text{T2P}}$ and $\mathfrak{D}_{\text{T2TL}}$ respectively. First, it creates a bootstrap dataset $\mathfrak{D}'$ from the original dataset $\mathfrak{D}$. This dataset comprises effective few-shot examples that are generated using rejection sampling. Since the bootstrapping process is done for both $\mathfrak{D}_{\text{T2P}}$ and $\mathfrak{D}_{\text{T2TL}}$, we can say $\mathfrak{D}' = \mathfrak{D}'_{\text{T2P}} \cup \mathfrak{D}'_{\text{T2TL}}$.

Next, *PULS* proposes k different instructions using an LLM depending on the properties of the original dataset $\mathfrak{D}$ and the original instruction, yielding the instruction sets $\mathbf{x}_{\text{T2P}} = \{x_{\text{T2P}_j}\}_{j=0}^{k}$ and $\mathbf{x}_{\text{T2TL}} = \{x_{\text{T2TL}_j}\}_{j=0}^{k}$. Thus, given a particular dataset entry i and instruction j:

$$\mathcal{P}^{\text{pred}}_{i,j} = \texttt{LLM}(\mathcal{T}_i, x_{\text{T2P}_j}, \theta_{llm}), \quad \mathcal{T}_i \in \mathfrak{D}'_{\text{T2P}} \tag{13}$$

$$\Phi^{\text{pred}}_{i,j} = \texttt{LLM}(\mathcal{T}_i, \mathcal{P}_i, x_{\text{T2TL}_j}, \theta_{llm}), \quad \mathcal{T}_i, \mathcal{P}_i \in \mathfrak{D}'_{\text{T2TL}} \tag{14}$$

Accuracy functions are defined as the following, where π is an evaluation metric that returns a similarity score between 0 and 1 of its inputs:

$$f_{\text{T2P}}(\mathcal{D}'_{\text{T2P}}, i, j) = \pi(\mathcal{P}_i, \mathcal{P}^{pred}_{i,j}), \quad \mathcal{P}_i \in \mathfrak{D}'_{\text{T2P}} \tag{15}$$

$$f_{\text{T2TL}}(\mathcal{D}'_{\text{T2TL}}, i, j) = \pi(\Phi_i, \Phi^{pred}_{i,j}), \quad \Phi_i \in \mathfrak{D}'_{\text{T2TL}} \tag{16}$$

Bayesian Optimization is used to create an optimal subset, $\mathfrak{D}_{\text{train}}$ of size $\mathcal{N}$, with the following operation:

$$\mathfrak{D}_{\text{T2P|train}} = \arg \max_{\mathfrak{D}_s \subset \mathfrak{D}'_{\text{T2P}}, |\mathfrak{D}_s| = \mathcal{N}} \left[\sum_{i \in \mathfrak{D}_s} \max_{0 \le j \le k} f_{\text{T2P}}(\mathfrak{D}_s, i, j) \right] \tag{17}$$

$$\mathfrak{D}_{\text{T2TL|train}} = \arg \max_{\mathfrak{D}_s \subset \mathfrak{D}'_{\text{T2TL}}, |\mathfrak{D}_s| = \mathcal{N}} \left[\sum_{i \in \mathfrak{D}_s} \max_{0 \le j \le k} f_{\text{T2TL}}(\mathfrak{D}_s, i, j) \right] \tag{18}$$

$$\mathfrak{D}_{\text{train}} = \mathfrak{D}_{\text{T2P|train}} \cup \mathfrak{D}_{\text{T2TL|train}}. \tag{19}$$

With this process, $\mathfrak{D}_{\text{train}}$ can be used as an effective few-shot dataset for *PULS*. LM_{T2P} uses $\mathfrak{D}_{\text{T2P|train}}$ and LM_{T2TL} uses $\mathfrak{D}_{\text{T2TL|train}}$ as their datasets. Prompt 1 and Prompt 2

detail the system prompts for the overall consistency mode in LM_{T2P} and LM_{T2TL} respectively.

PULS Prompt for Text to Propositions (.md)

```
**System Message**:
Your input fields are:
1. 'input_prompt' (str): Input prompt
summarizing what happened in a video.
Your output fields are:
1. 'reasoning' (str)
2. 'output_propositions' (str): A list
of atomic propositions that correlate with
the inputted prompt. For example, for a
prompt such as 'A person holding a hotdog is
walking down to the street where many cars
next to the huge truck', the propositions are
'person holds hotdog', 'person walks', and
'car next to truck'. This outputted list of
propositions MUST be formatted as: [prop1,
prop2, prop3].
Your objective is:
Convert from a prompt to a list of
propositions using the following schema.

---

**User message**:
Input Prompt: A boat sailing leisurely along
the Seine River with the Eiffel Tower in
background, zoom out.
Respond with the corresponding output fields,
starting with the field 'reasoning', then
'output_propositions'.

**Assistant message**:
Reasoning: Not supplied for this particular
example.
Output Propositions: ['There is a boat', 'The
boat is sailing leisurely', 'The boat is along
the Seine River', 'The Eiffel Tower is in the
background', 'The view is zooming out']

...
< redacted more examples >
...
```

Prompt 1. **T2P Prompt for *PULS*.** System prompt to map prompts to the propositions for the "Overall Consistency" mode. Other modes follow a similar structure, albeit with minor changes in instructions and few-shot examples.

11. Vision Language Model Calibration

In this section, we provide the implementation details to detect the existence of propositions obtained by *PULS* to label each frame in the synthetic video. First, we describe how VLMs conduct inference and then describe the methodology for obtaining the optimal thresholds for calibrating the vision language model.

11.1. Inference Via Vision Language Models

We use VLM as a semantic detector as explained in Section 4.2. We pass each atomic proposition $p_i \in \mathcal{P}$ such as "person", "car", "person in the car", etc. along with Section 11.1. Once the VLM outputs either 'Yes' or 'No', we compute the token probability of the response and use it as a confidence score for the detection.

PULS Prompt for Text to Temporal Logic (.md)

```
**System Message**:
Your input fields are:
1. 'input_prompt' (str): Input prompt
summarizing what happened in a video.
2. 'input_propositions' (str): A list of
atomic propositions in the video.
Your output fields are:
1. 'reasoning' (str)
2. 'output_specification' (str): The formal
specification of the event sequence. The
response will be a linear temporal logic (LTL)
formula made of the inputted propositions
combined with LTL symbols. Each symbol can
only be one of ['AND', 'OR', 'NOT', 'UNTIL',
'ALWAYS', 'EVENTUALLY'].
Your objective is:
Convert from a prompt and a list of
propositions to a TL specification using
the following schema.

---

**User message**:
Input Prompt: A baseball glove on the right
of a tennis racket, front view.
Input Propositions: ['There is a baseball
glove', 'There is a tennis racket', 'The
baseball glove is on the right of the tennis
racket', 'The view is from the front']
Respond with the corresponding output fields,
starting with the field 'reasoning', then
'output_specification'.

**Assistant message**:
Reasoning: Not supplied for this particular
example.
Output Specification: (There is a baseball
glove AND There is a tennis racket AND The
baseball glove is on the right of the tennis
racket AND The view is from the front)

...
< redacted more examples >
...
```

Prompt 2. **T2TL Prompt for *PULS*.** System prompt to map prompts and propositions for the "Overall Consistency" mode. Other modes follow a similar structure, albeit with minor changes in instructions and few-shot examples.

Prompt for Semantic Detector (VLM)

```
Is there {atomic proposition (p_i)} present in
the sequence of frames?
[PARSING RULE] 1.  You must only return a Yes
or No, and not both, to any question asked.
2.  You must not include any other symbols,
information, text, or justification in your
answer or repeat Yes or No multiple times.
3.  For example, if the question is 'Is there
a cat present in the Image?', the answer must
only be 'Yes' or 'No'.
```

Prompt 3. **Semantic Detector VLM.** Used to identify the atomic proposition within the frame by initiating VLM with a single frame or a series of frames.

11.2. False Positive Threshold Identification

Dataset for Calibration: We utilize the COCO Captions [7] dataset to calibrate the following open-source vision language models – InternVL2 Series (1B, 2B, 8B) [45] and LLaMA-3.2 Vision Instruct [2] – for *NeuS-V*. Given that each image-caption pair in the dataset is positive coupling, we construct a set of negative image-caption pairs by randomly pairing an image with any other caption corresponding to a different image in the dataset. Once we construct the calibration dataset, which comprises 40000 image caption pairs, we utilize the VLM to output a 'Yes' or a 'No' for each pair.

Thresholding Methodology We can identify the optimal threshold for the VLM by treating the above problem as either a single-class or multi-class classification problem. We opt to do the latter. The process involves first compiling detections into a list of confidence scores and one-hot encoded ground truth labels. We then sweep through all available confidence scores to identify the optimal threshold. Here, we calculate the proportion of correct predictions by applying each threshold (see Figure 7). The optimal threshold is identified by maximizing accuracy, which is the ratio of the true positive and true negative predictions. Additionally, to comprehensively evaluate model behavior, we compute Receiver Operating Characteristics (ROC) as shown in Figure 7, by computing the true positive rate (TPR), and false positive rate (FPR) across all thresholds. Once we obtain the optimal threshold, we utilize it to calibrate the predictions from the VLM. We show the accuracy vs confidence plots before and after calibration in Figure 7.

12. Video Automaton Generation Function

Given a calibrated score set (see Equation (20)) across all frames $\mathcal{F}_n$ (where n is the frame index of the video) and propositions in $\mathcal{P}$, we construct the video automaton $\mathcal{A}_\mathcal{V}$ using the video automaton generation function (see Equation (7)).

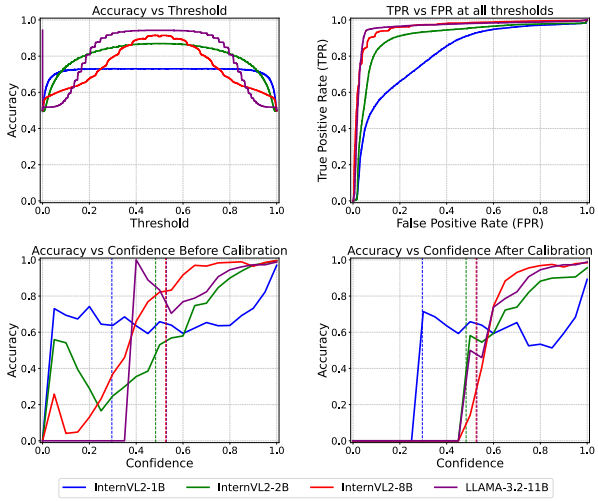

Figure 7. **Calibration Plots.** We plot the accuracy vs threshold for all VLMs on our calibration dataset constructed from the COCO Caption dataset (top left). We plot the True Positive Rate (TPR) vs False Positive Rate (FPR) across all thresholds on the top right. Finally, the bottom plots show the confidence vs accuracy of the model before and after calibration, respectively.

$$C^\star = \{C^\star_{p_i,j} \mid p_i \in \mathcal{P}, j \in \{1, 2, \ldots, n\}\}. \quad (20)$$

As shown in Algorithm 3, we first initialize the components of the automaton, including the state set Q, the label set λ, and the transition probability set δ, all with the initial state q_0. Next, we iterate over $C^\star$, incrementally constructing the video automaton by adding states and transitions for each frame. This process incorporates the proposition set and associated probabilities of all atomic propositions. We compute possible labels for each frame as binary combinations of $\mathcal{P}$ and calculate their probabilities using the $C^\star$.

13. *NeuS-V* Prompt Suite

Creating the Dataset: Our dataset is carefully designed to evaluate temporal fidelity and event sequencing in generated videos. It spans four themes – "Nature", "Human & Animal Activities", "Object Interactions", and "Driving Data", with each theme containing 40 prompts. These prompts vary in complexity, categorized into 20 basic, 15 intermediate, and five advanced prompts based on the number of temporal and logical operators (see Table 5). These prompts were generated using GPT-4o with the system prompt in Prompt 4. To ensure quality, each prompt was manually verified for clarity, completeness, and relevance.

Generating Videos from Prompts: We generated videos for all 160 prompts using both open-source and closed-source models. For open-source models, we utilized pre-trained weights and code by cloning their HuggingFace Spaces and querying them using `huggingface_client`. For closed-source models (Gen3 and Pika), we implemented a custom CLI that reverse-engineers their front-end interfaces to automate video generation requests. In total, we produced 640 videos (160 prompts $\times$ four models). We also plan to launch a publicly available leaderboard on HuggingFace after acceptance, which will allow continuous evaluation of new T2V models as they emerge.

Annotations: Annotations were crowdsourced from 20 participants, including contributors from social media platforms like X (formerly Twitter) and LinkedIn. As shown in Figure 8, annotators were instructed to evaluate the alignment of videos with their corresponding prompts while disentangling visual quality from text-to-video alignment.

Insights into Prompts: We showcase representative examples from our dataset in Table 6 and Table 7, highlighting the diversity and complexity of prompts. These examples provide prompts to represent them in different modes. In the future, we plan to expand the dataset by adding more prompts across existing themes and introducing new categories to further enhance its utility and scope.

Theme	Complexity			Total Prompts
	Basic	Intermediate	Advanced	
Nature	20	15	5	40
Human & Animal Activities	20	15	5	40
Object Interactions	20	15	5	40
Driving Data	20	15	5	40
Total	**80**	**60**	**20**	**160**

Table 5. **Statistics of *NeuS-V* Prompt Suite.** We include prompts from various themes across different complexities to evaluate T2V models on a total of 160 prompts.

Algorithm 3: Video Automaton Generation

Input: Set of semantic score across all frames given all atomic propositions $\{C^\star = C^\star_{p_i,j} \mid p_i \in \mathcal{P}, j \in \{1, 2, \ldots, n\}\}$, set of atomic propositions $\mathcal{P}$

Output: Video automaton $\mathcal{A}_\mathcal{V}$

begin
- $Q \leftarrow \{q_0\}$ // Initialize the set of states with the initial state
- $\lambda \leftarrow \{(q_0, \text{initial})\}$ // Initialize the set of labels with the initial label
- $\delta \leftarrow \{\}$ // Initialize the set of state transitions
- $Q_p \leftarrow \{q_0\}$ // Track the set of previously visited states
- $n \leftarrow \frac{|C^\star|}{|\mathcal{P}|}$ // Calculate the total number of frames n
- **for** $j \leftarrow 1$ **to** n **do**
  - $Q_c \leftarrow \{\}$ // Track the set of current states
  - **for** $e^k_j \in 2^{|\mathcal{P}|}$ **do**
    - // e^k_j: unique combination of 0s and 1s for atomic propositions in $\mathcal{P}$
    - $\lambda(q^k_j) = \{v_1, v_2, \ldots, v_i \mid v_i \in \{1, 0\}, \forall i \in \{1, 2, \ldots, |\mathcal{P}|\}\}$
    - $pr(j, k) \leftarrow 1$ // Initialize probability for the label
    - **for** $v_i \in \lambda(q^k_j)$ **do**
      - // Calculate probability for e^k_j
      - **if** $v_i = 1$ **then**
        - $pr(j, k) \leftarrow pr(j, k) \cdot C^\star_{p_i,j}$
      - **else**
        - $pr(j, k) \leftarrow pr(j, k) \cdot (1 - C^\star_{p_i,j})$
      - // Add state and define transitions if the probability is positive
      - **if** $pr(j, k) > 0$ **then**
        - $Q \leftarrow Q \cup \{q^k_j\}$
        - $Q_c \leftarrow Q_c \cup \{q^k_j\}$
        - $\lambda \leftarrow \lambda \cup \{(q^k_j, \lambda(q^k_j))\}$
        - **for** $q_{j-1} \in Q_p$ **do**
          - $\delta(q_{j-1}, q^k_j) \leftarrow pr(j, k)$
          - $\delta \leftarrow \delta \cup \{\delta(q_{j-1}, q^k_j)\}$
        - **end for**
      - **end for**
    - $Q_p \leftarrow Q_c$ // Update previous state
    - **end for**
  - **end for**
- // Add terminal state
- $Q \leftarrow Q \cup \{q^0_{j+1}\}$
- $\lambda \leftarrow \lambda \cup \{(q^0_{j+1}, \text{terminal})\}$
- **for** $q_{j-1} \in Q_p$ **do**
  - $\delta(q_{j-1}, q^0_{j+1}) \leftarrow 1$
  - $\delta \leftarrow \delta \cup \{\delta(q_{j-1}, q^0_{j+1})\}$
- **end for** // Return video automaton
- $\mathcal{A}_\mathcal{V} \leftarrow (Q, q_0, \delta, \lambda)$
- **return** $\mathcal{A}_\mathcal{V}$

Generating Temporally Extended Prompts (.md)

```
**Objective**: Generate individual prompts for a
text-to-video generation benchmark. Each prompt
should focus on specific temporal operators and
adhere to the given theme and complexity level.
Your goal is to create clear, vivid prompts
that illustrate events occurring in time, with a
strong emphasis on temporal relationships.

---

#### **Instructions for Prompt Generation**
1. **Theme**: One of the following themes:
Nature, Human and Animal Activities, Object
Interactions, or Driving Data
2. **Complexity Level**:
   - **Basic (1 Operator)**: Use only **one
   temporal operator** ("Always," "And," or
   "Until") in the prompt.
   - **Intermediate (2 Operators)**: Use **two
   temporal operators** in a sequence. The
   prompt should clearly connect the events with
   each operator in a natural, coherent way.
   - **Advanced (3 Operators)**: Use **three
   temporal operators** in a chain, showing a
   progression of events. Each part should flow
   logically to the next.
3. **Available Temporal Operators**:
   - **"Always"**: Describes an event that
   continuously occurs in the background or
   context.
   - **"And"**: Combines two events happening
   simultaneously or in coordination.
   - **"Until"**: Describes an event that
   occurs until another event starts.

---

#### **Examples**
##### **Theme: Nature**
- **Basic (1 Operator)**:
   - "Always a river flowing gently."
   - "Rain pouring until the sun comes out."
   - "A bird chirping and a dog barking nearby."

            ...
< redacted more examples >
            ...
---

**Ensure** that each prompt uses only the
specified number of operators based on
complexity, with clear temporal transitions
between events. Use vivid language to create
a realistic and engaging scenario. Try not to
use language that is too abstract or ambiguous,
focusing on concrete actions and events. Do not
include any acoustic information in the prompts,
as they are meant to describe visual scenes only.

I shall provide you with a theme, complexity
level, and the number of prompts you need to
generate. You must only output the prompts,
each on a new line, without any additional
information.

Are you ready?
```

Prompt 4. **System Prompt for *NeuS-V* Prompt Suite.** Used to query GPT-4o to generate temporally extended prompts across different themes and complexities.

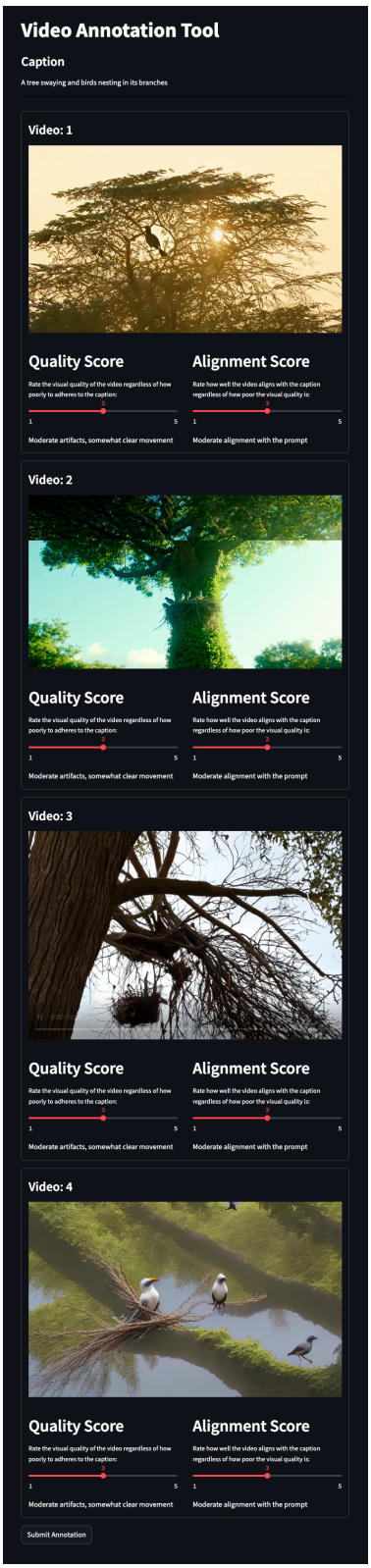

Figure 8. **Tool for Annotating Videos.** Subjects are instructed to disambiguate quality and alignment during annotation, scoring each from a range of 1 through 5.

Theme	Complexity	Prompts and Specification Modes
Nature	Basic	**Prompt:** Snow falling until it covers the ground **Object Existence:** (“snow”) U (“ground”) **Object Action Alignment:** (“snow_falls” U “ground_is_covered”) **Spatial Relationship:** F (“snow_covers_ground”) **Overall Consistency:** (“snow_falling” U “it_covers_the_ground”)
		Prompt: Always waves crashing against the rocky shore **Object Existence:** G (“waves” & “shore”) **Object Action Alignment:** G (“waves_crash_against_rocky_shore”) **Spatial Relationship:** G (“waves_on_shore”) **Overall Consistency:** G (“waves_crashing_against_the_rocky_shore”)
	Intermediate	**Prompt:** The sun shining until the clouds gather, and then rain begins to fall **Object Existence:** (“sun_shining” U “clouds_gather”) & F (“rain_begins_to_fall”) **Object Action Alignment:** (“sun_shines” U “clouds_gather”) & F (“rain_falls”) **Spatial Relationship:** G (“sun_over_horizon” & “dew_on_grass”) **Overall Consistency:** (“sun_shining” U “clouds_gather”) & F (“rain_begins_to_fall”)
		Prompt: A butterfly resting on a flower until a gust of wind comes, and then it flies away **Object Existence:** (“butterfly” & “flower”) U (“wind”) **Object Action Alignment:** (“butterfly_rests_on_flower” U “gust_of_wind_comes”) & F (“butterfly_flies_away”) **Spatial Relationship:** (“butterfly_on_flower”) U (“butterfly_flies_away”) **Overall Consistency:** (“butterfly_resting_on_a_flower” U “gust_of_wind_comes”) & F (“it_flies_away”)
	Advanced	**Prompt:** Always a river flowing quietly through the valley, until the sky darkens with storm clouds, and rain begins to pour heavily **Object Existence:** G (“river” & “storm_clouds”) **Object Action Alignment:** ((“river_flows_quietly” U “sky_darkens”)) & F(“rain_pours”) **Spatial Relationship:** G(“river_flowing_through_valley”) U (“sky_darkens_with_storm_clouds”) & F “rain_begins_to_pour_heavily” **Overall Consistency:** G ((“river_flowing_quietly_through_the_valley”) U “sky_darkens_with_storm_clouds”) & F “rain_begins_to_pour_heavily”
Human and Animal Activities	Basic	**Prompt:** A dog barking until someone throws a ball **Object Existence:** (“dog”) U (“ball”) **Object Action Alignment:** (“dog_barks” U “someone_throws_ball”) **Spatial Relationship:** G (“clock_on_mantle” & “fireplace_beneath_mantle”) **Overall Consistency:** (“dog_barking” U “someone_throws_a_ball”)
		Prompt: A bird singing until it flies away to another branch **Object Existence:** (“bird”) U (“branch”) **Object Action Alignment:** (“bird_sings” U “bird_flies_away_to_another_branch”) **Spatial Relationship:** (“bird_on_branch”) U (“bird_on_different_branch”) **Overall Consistency:** (“bird_singing” U “it_flies_away_to_another_branch”)
	Intermediate	**Prompt:** A child building a sandcastle until the tide rises, and then they watch it wash away **Object Existence:** (“child” & “sandcastle”) U “tide” **Object Action Alignment:** ((“child_builds_sandcastle” U “tide_rises”)) & F(“child_watches_it_wash_away”) **Spatial Relationship:** (“child_building_sandcastle” U “tide_rises”) **Overall Consistency:** (“child_building_a_sandcastle” U “tide_rises”) & F “child_watches_sandcastle_wash_away”
		Prompt: Always a cat lounging on the porch, and butterflies fluttering around **Object Existence:** G (“cat” & “butterflies”) **Object Action Alignment:** G((“cat_lounges” & “butterflies_flutter”)) **Spatial Relationship:** G(“cat_lounging_on_porch” & “butterflies_fluttering_around”) **Overall Consistency:** G (“cat_lounging_on_the_porch” & “butterflies_fluttering_around”)
	Advanced	**Prompt:** A dog digging in the backyard, until its owner arrives, and then they play fetch together **Object Existence:** (“dog” & “backyard”) U (“owner” & “ball”) **Object Action Alignment:** (“dog_digs_in_backyard” U “owner_arrives”) & F (“dog_plays_fetch_with_owner”) **Spatial Relationship:** (“dog_in_backyard”) U (“owner_arrives”) **Overall Consistency:** (“dog_digging_in_the_backyard” U “its_owner_arrives”) & F (“they_play_fetch_together”)

Table 6. ***NeuS-V* Prompt Suite:** Illustrative prompts and their detailed specifications (across all four modes) for varying complexities within the “Nature” and “Human and Animal Activities” themes.

Theme	Complexity	Prompts and Specification Modes
Object Interactions	Basic	**Prompt:** A lamp glowing until it is turned off **Object Existence:** (“lamp”) U (¡ ‘lamp”) **Object Action Alignment:** (“lamp_glows” U “lamp_is_turned_off”) **Spatial Relationship:** (“cars_passing_by_person”) **Overall Consistency:** (“lamp_glowing” U “it_is_turned_off”)
		Prompt: A car engine running and the dashboard lights flashing **Object Existence:** (“car_engine” & “dashboard_lights”) **Object Action Alignment:** (“car_engine_runs”&“dashboard_flashes”) **Spatial Relationship:** (“car_engine_running” & “dashboard_lights_flashing”) **Overall Consistency:** (“car_engine_running” & “dashboard_lights_flashing”)
	Intermediate	**Prompt:** Always a record player spinning a vinyl, and light glowing softly from a nearby lamp **Object Existence:** G (“record_player” & “lamp”) **Object Action Alignment:** G((“record_player_spins”&“lamp_glows”)) **Spatial Relationship:** G(“vinyl_on_record_player” & “light_glowing_from_nearby_lamp”) **Overall Consistency:** G (“record_player_spinning_a_vinyl” & “light_glowing_softly_from_a_nearby_lamp”)
		Prompt: A drone hovering in the air until it reaches its next waypoint, and then it continues to fly **Object Existence:** (“drone”) U (“waypoint”) **Object Action Alignment:** (“drone_hovers_in_air” U “drone_reaches_next_waypoint”) & F (“drone_continues_to_fly”) **Spatial Relationship:** (“drone_in_air”) U (“drone_reaches_waypoint”) **Overall Consistency:** (“drone_hovering_in_air” U “drone_reaches_next_waypoint”) & F (“drone_continues_to_fly”)
	Advanced	**Prompt:** A lightbulb flickering intermittently, until the switch is turned off, and then the room is cast into darkness **Object Existence:** (“lightbulb”) U “switch” **Object Action Alignment:** ((“lightbulb_flickers” U “switch_turned_off”)) & F(“room_darkens”) **Spatial Relationship:** (“lightbulb_flickering” U “switch_is_turned_off”) **Overall Consistency:** (“lightbulb_flickering_intermittently” U “switch_is_turned_off”) & F “room_is_cast_into_darkness”
Driving Data	Basic	**Prompt:** The vehicle moving forward until it reaches a stop sign **Object Existence:** “vehicle” U “stop_sign” **Object Action Alignment:** (“vehicle_moves” U “vehicle_reaches_stop_sign”) **Spatial Relationship:** (“vehicle_moving_forward” U “vehicle_at_stop_sign”) **Overall Consistency:** (“vehicle_moving_forward” U “vehicle_reaches_a_stop_sign”)
		Prompt: A motorcycle revving and a bus pulling up beside it **Object Existence:** (“motorcycle” & “bus”) **Object Action Alignment:** (“motorcycle_revs”&“bus_pulls_up”) **Spatial Relationship:** (“motorcycle_revving” & “bus_pulling_up_beside”) **Overall Consistency:** (“motorcycle_revving” & “bus_pulling_up_beside”)
	Intermediate	**Prompt:** A traffic light turning red until pedestrians finish crossing, and then it shifts to green **Object Existence:** (“traffic_light”) U “pedestrians” **Object Action Alignment:** ((“traffic_light_is_red” U “pedestrians_finish_crossing”)) & F(“traffic_light_turns_green”) **Spatial Relationship:** (“traffic_light_turning_red” U “pedestrians_finish_crossing”) **Overall Consistency:** (“traffic_light_turning_red” U “pedestrians_finish_crossing”) & F “traffic_light_shifts_to_green”
		Prompt: Always an electric vehicle charging at the station, and its driver reading a book nearby **Object Existence:** G (“vehicle” & “driver”) **Object Action Alignment:** G((“electric_vehicle_charges”&“driver_reads”)) **Spatial Relationship:** G(“electric_vehicle_charging_at_station” & “driver_reading_book_nearby”) **Overall Consistency:** G (“electric_vehicle_charging_at_the_station” & “driver_reading_a_book_nearby”)
	Advanced	**Prompt:** A car driving through the city streets, until it encounters a construction zone, and then it reroutes to an alternate path **Object Existence:** (“car” & “streets”) U (“construction_zone” & “path”) **Object Action Alignment:** (“car_drives_through_city_streets” U “car_encounters_construction_zone”) & F (“car_reroutes_to_alternate_path”) **Spatial Relationship:** (“car_on_city_streets”) U (“car_at_construction_zone”) **Overall Consistency:** (“car_driving_through_city_streets” U “it_encounters_a_construction_zone”) & F (“it_reroutes_to_an_alternate_path”)

Table 7. ***NeuS-V* Prompt Suite (continued):** Illustrative prompts and their detailed specifications (across all four modes) for varying complexities within the “Object Interactions” and “Driving Data” themes.