Enhancing Vision Language Models with Logic Reasoning for Situational Awareness

Pavana Pradeep Kumar[1], Krishna Kant[1] and Suya You[2]

***Abstract*—Vision Language Models (VLMs) can provide lucid high-level descriptions of complex activities from images and videos and thus form a powerful tool for situational awareness (SA) applications where we are primarily interested in unusual/anomalous events. Therefore, the SA applications require high reliability in recognizing such events, even though they typically have limited training instances. Furthermore, we also need a means to indicate the quality of the result. In this paper, we integrate VLMs with traditional computer vision techniques through the use of explicit logic reasoning to enhance situational awareness: (a) a mechanism to intelligently select inputs for fine-tuning of VLMs for infrequent events without requiring labeled data, (b) An intelligent fine-tuning mechanism that yields much better accuracy than an uninformed selection, and (c) a mechanism to produce justification of VLM output during the inferencing phase. We demonstrate that our intelligent fine-tuning mechanism improves the accuracy and provides a valuable means, during inferencing, to either confirm the validity of the VLM output or indicate why it may be questionable.**

***Impact Statement*—This paper applies emerging Vision Language Models (VLMs) for vision-based situational awareness in cyber-physical environments, focusing on safety, security, and policy compliance. While VLMs offer high-level descriptions of infrequent but critical events, traditional computer vision (TCV) methods better capture finer details like people, objects, locations, and movements. We explore integrating VLM and TCV through logical reasoning to enhance situational awareness, improve fine-tuning efficiency, and ensure output reliability. Fine-tuning VLMs can be costly, especially for infrequent events. We propose a technique that not only makes the fine-tuning efficient but also offers a way for sanity checking of the outputs during inferencing. The latter ensures enhanced reliability in recognizing crucial events, which is critical in the situational awareness context.**

***Index Terms*—Multimodal Language Models, Logical Reasoning, Anomalous Activity Detection, Video-based Monitoring, Situational Awareness**

I. Introduction

With video-based monitoring of various cyber-physical systems becoming ubiquitous, a crucial need is to perform automated situational understanding using the streaming data from the deployed video cameras and possibly other sensors. The essential purpose of such a mechanism is to recognize situations that may be anomalous in some way. Typically, these concerns include safety, security, policy violations, or other unusual events. For example, in traffic monitoring on the road, it may be desirable to detect accidents, near-accidents, criminal activities using vehicles, etc. These situations are less common than normal traffic but of primary interest in monitoring. We assume throughout this paper that the specific activities/events of interest for a given situational awareness application are known (or can be learned) and can be characterized at least approximately. We henceforth denote the set of these "main" activities as $\mathcal{A}^m$.[1]

Vision Language Models (VLMs) have recently burst onto the scene with impressive abilities to summarize the content of an image or short video at an advanced level. Most VLMs use a Large Language Model (LLM) backend, which enables them to engage in a Q&A capability and lucid descriptions of what is happening in the image/video. These descriptions go well beyond what is reasonably achievable using Traditional Computer Vision (TCV) techniques [50]. Here, TCV means deep learning networks such as CNNs specifically designed and trained for purposes like recognizing objects, object attributes, specific actions, etc.

However, the TCV techniques remain important and can be exploited in multiple ways, especially in a situational awareness context. First, in addition to a high-level description given by VLMs for activities in the set $\mathcal{A}^m$, we also want to know the finer details, such as the relative locations and movements of people and machinery for each important activity/event. Fine details can be easily obtained by recognizing objects and their poses/attributes via TCV techniques and tracking their locations/movements from frame to frame. Such details are difficult to obtain directly from the VLM. Thus, an integration of VLM and TCV can provide both the high-level context/activities and the low-level details associated with $\mathcal{A}^m$. The "glue" in this integration is the logical reasoning techniques that can run very efficiently in real-time.

The second way to exploit TCV is in fine-tuning the VLMs, which is essential to achieve good performance for the desired set $\mathcal{A}^m$ [11]. The accuracy of an out-of-the-box VLM could vary widely across the activities in $\mathcal{A}^m$, and it is important to intelligently select the images/videos for fine-tuning containing activities where the VLM is performing poorly. One reason for doing this is the rather high resource requirements for fine-tuning VLMs. The second reason is that the activities in $\mathcal{A}^m$ typically occur infrequently. Thus, it is important to (a) identify video segments containing activities from $\mathcal{A}^m$ quickly and automatically, even if the identification is only approximate, and (b) evaluate the subset of $\mathcal{A}$ where the VLM is performing poorly before proceeding with the task of labeling video segments and using them for further fine-tuning.

[1]CIS Department, Temple University, Philadelphia, PA, USA. (e-mail: pavana.pradeep@temple.edu, kkant@temple.edu)

[2]ARL, San Jose, CA. (e-mail: Suya.you.civ@army.mil)

[1]The reason for using the word "main" and the superscript m shall be clear shortly.

For (a), we exploit TCV, which typically can go through large amounts of data quickly to identify the segments for selection. The idea is to identify a set of *proxy activities*, say $\mathcal{A}^p$, that can approximately capture the activities in $\mathcal{A}^m$ but are simple enough to be recognized easily by TCV. This identification can itself be automated, as we discuss later. For (b), we identify an *auxiliary activity set* $\mathcal{A}^a$, which contains a variant of every activity in $\mathcal{A}^m$. Thus, we shall make use of two VLMs, a *main VLM* (VLM^m) and an auxiliary VLM (VLM^a). Inconsistencies between the outputs of these VLMs or with the proxy activity recognition can then pinpoint the need for further fine-tuning.

While the use of an additional VLM may appear to add substantial overhead to fine-tuning, we demonstrate that it still outperforms the uninformed fine-tuning. Furthermore, it provides two other key advantages: (a) The evaluation is now based on consistency rather than accuracy and hence does not need labeled video segments, and (b) we can continue to use the mechanism during the inferencing phase as well and thus have a strong, strong justification mechanism. Both are very significant advantages in situational awareness because while the events in $\mathcal{A}^m$ may be infrequent/rare, recognizing them correctly is crucial. Again, logical reasoning plays a key role in enabling these functionalities.

To the best of our knowledge, this is the first work of its type to integrate explicit logic reasoning with VLM and TCV to make three substantial contributions in the context of VLM-based situational awareness: (a) a mechanism to intelligently select inputs for fine-tuning of VLMs for infrequent events without requiring labeled data, (b) An intelligent fine-tuning mechanism that yields much better accuracy than an uninformed selection even when the length of the fine-tuning period is kept the same, and (c) a mechanism to produce justification of VLM output during the inferencing phase.

As VLMs become mainstream and are further optimized for speed (e.g., Video-Mamba), the techniques proposed in this paper and their further enhancements will be crucial to using VLMs in real-time situational awareness applications confidently.

The rest of this paper is organized as follows. Section II presents the detailed design of our directed fine-tuning mechanism. Section IV discusses the experimental assessment of the mechanism. Section VI discusses the related work. Finally, section VII concludes the discussion.

II. Proposed Framework

A. Tasks and Activities

As stated in the last section, we target the recognition of a set of main activities $\mathcal{A}^m$ for the desired application, collectively referred to as the *Main task* or T^m. We assume that we have I targeted activities, i.e., $\mathcal{A}^m = (A^m_1, ..., A^m_I)$. We also have the *Auxiliary task*, or T^a with $\mathcal{A}^a = (A^a_1, ..., A^a_I)$, with 1-to-1 correspondence between A^m_i and A^a_i's. We discuss later in section III-A how to automatically derive A^a_i from A^m_i. We also have our main and auxiliary VLMs, VLM^m and VLM^a, to recognize A^m_i and A^a_i respectively as belonging to "class" i. It is certainly possible to use the power of LLMs to go beyond a direct class-based output from VLMs, but we leave that to future work.

Task type	**Defined via**
Main Task	Main activity set ($\mathcal{A}^m = \{A^m_i, i = 1..I\}$)
Auxiliary Task	Auxiliary activity set ($\mathcal{A}^a = \{A^a_i, i = 1..I\}$)
Proxy Task	Proxy activity set ($\mathcal{A}^p = \{A^p_k, k = 1..K\}$)

TABLE I: Task and Activities

We also introduce the *proxy task*, denoted as T^p that recognizes the proxy activity set $\mathcal{A}^p = (A^p_1, ..., A^p_K)$ for some $K > 1$. Proxy activities are defined as simpler components of main/auxiliary activities that can be recognized by standard object recognition or related algorithms. Table I summarizes this terminology for easy reference.

B. Defining and Exploiting Proxy Activities

Fig. 1 further depicts the relationship between tasks and activities. Although this figure refers to the main task, a similar description will apply to the auxiliary task as well. It shows that task T^m consists of activities A^m_i, each of which is associated with some proxy activities, henceforth denoted as $\mathbb{S}^m_i(A^p)$. For example, A^m_2 has three associated proxy activities $A^p_1..A^p_3$.

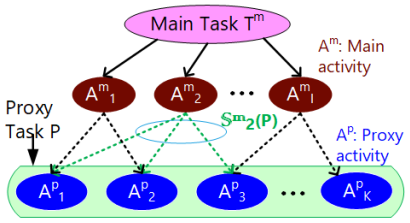

Fig. 1: Task and Activity Relationship

We now place the requirement that $A^m_i \Longrightarrow \mathbb{S}^m_i(A^p)$. Here, $\Longrightarrow$ means that if the activity A^m_i is observed, all of the (simpler) activities in the set $\mathbb{S}^m_i(A^p)$ should also be observed. That is, the set $\mathbb{S}^m_i(A^p)$ is necessary for A^m_i to occur but may not be sufficient. The exposition here is informal; in actual implementation, such requirements and the underlying activities are represented as *logic assertions* as discussed later.

The intent is to use TCV to recognize $\mathbb{S}^m_i(A^p)$ and exploit the one-way implication as a weak but efficient consistency check with the VLM output. For TCV, we use standard algorithms such as YOLO variants to detect objects/poses, potentially augmented with additional neck/head layers for recognizing additional attributes if needed [21]. Any movement tracking can be done by frame-to-frame monitoring of objects, perspective transformations, and reasoning about the locations/movements of objects. In particular, if a person/machine disappears from view temporarily and returns, we can track it easily and accurately in most cases. For simplicity, we also use the same set of proxy activities for the auxiliary VLM. That is, $A^a_j \Longrightarrow \mathbb{S}^a_j(A^p)$, where $\mathbb{S}^a_j(A^p)$ is some non-null subset of A^p.

It is important to note that we *do not* need to assume that TCV recognition is always correct since we are only looking for consistency between TCV and VLM outputs. By their simplicity and mature/focused TCV recognition algorithms, we expect the proxy actions to be recognized quite accurately. Furthermore, with more effort, we can catch mistakes by TCV

algorithms by exploiting the key properties of consistency and smoothness of objects and actions across frames. Such properties, too, can be easily formulated using logical reasoning, although we do not pursue this line in this paper.

C. Consistency Driven Fine-tuning

Given the setup above, we can design an algorithm to process the video frames through VLM^m, VLM^a, and TCV, and determine, via spatio-temporal logic reasoning (a) Whether the VLM^m output class, say i is same as the VLM^a output class, say i', (b) Whether all the activities in the set $\mathbb{S}_i^m(A^p)$ are observed and (c) Whether all the activities in the set $\mathbb{S}_{i'}^a(A^p)$ are also observed. If any of these *consistency conditions* do not hold, we conduct an intelligent or **directed** selection of (or "inputs") for further fine-tuning of the VLM^m and VLM^a.

The process can be repeated in a loop until the fine-tuning performance saturates or is terminated for other reasons (e.g., running out of fine-tuning data). We henceforth call the data used to evaluate fine-tuning needs as *evaluation data* or simply *eval-data*. Note that our evaluation is purely consistency-driven and does not need labeled data. We shall contrast this with the traditional *accuracy driven* evaluation, which does need labeled data. We show that the *consistency-driven* evaluation is on par with the *accuracy-driven* evaluation and thus can be used easily to judge if further fine-tuning is needed as the environment evolves.

D. Justifying Inferences

Because of the inherent inaccuracies in any automated video processing, a real-world deployment must justify the output or indicate when the output might be unreliable. The proposed *consistency-driven* fine-tuning mechanism can be adapted to this role by simply continuing the consistency checks during the inference phase. In this case, we could run the two VLMs in parallel on two different GPUs, if available, or forego VLM^a during inferencing and thus weaken the validation. A third choice is to use VLM^a only for "difficult scenarios". Such scenarios can be learned from the history of observed inconsistencies, but we do not pursue that aspect here. We later quantify the overhead of justification, which should be possible to do in real time with improvements in hardware speeds.

III. Implementing Directed Fine Tuning

A. Identifying Proxy and Auxiliary Activities

In this section, we discuss the issue of automated identification of suitable proxy and auxiliary activities related to a primary activity. We start with proxy activities, where we need repertories of generic and easily recognized activities/events relevant to the application. In most cases, these include spatial relationships, poses/pose-transitions of people, everyday actions for people and robots like walking, sitting, raising arms, etc., relevant movements of machinery, and others. The TCV algorithm and direct frame processing should be capable of recognizing all such activities in the repertoire. For example, the object detector recognizes two objects (e.g., a pedestrian and a car). In contrast, the distance between them, speed, direction of movement, etc., can determined by a direct frame analysis, assuming that the camera position is known.

We next automatically determine the mapping $\mathbb{S}_I(A^p)$ for each A_i, i.e., the set of proxy activities implied by the VLM recognized activity A_i. (We have omitted the superscript m or a since the same procedure applies to both.) First, we express every proxy activity A^p as a logic assertion, say ψ_{A^p}. Let D denote the entire input dataset available for fine-tuning. This would be a labeled set of short video segments (or images, if image-based VLM is used.) Then, for each class i, we pass each input through TCV and identify which assertions in ψ_{A^p} hold for it. In the last step, for each class i, we pick the assertions that hold in most (e.g., 90%) cases.[2] This set then characterizes the set $\mathbb{S}_i$ for each i.

To obtain the auxiliary activity A_i^a corresponding to the given A_i^m, we can exploit the VLM technology by asking for an alternate description of the activity. For independence, using a different VLM such as Video-chatGPT [29] is useful. The important point is to (a) accept only descriptions for which the identified set of proxy activities and (b) ensure that the meaning of the generated variant A_i^a is closely aligned with that of the original A_i^m. This can be checked easily using the mechanism in [29], which provides several measures to determine how well the two meanings are aligned (using a scale of 0-5). The scores include correctness, consistency across different responses, contextual understanding, and understanding of temporal changes. Only the first two are really relevant to our case.

TABLE II: Description of Scores

Average measure of performance	Score
Correctness score	4.09
Detailed orientation score	3.91
Contextual understanding score	3.97
Temporal understanding score	4.01
Consistency score	4.02

Table II shows the results of these scores describing how close the VLM generated auxiliary activity, A_i^a, is closely aligned with that of the original A_i^m. It is clear from the results that the VLM^m and VLM^a activities are very well aligned (a score of 4.0 or higher is considered to be very good [29]).

B. Using Explicit Logic Reasoning

Explicit logic reasoning has been used successfully in numerous domains [1] and contexts [32]–[35]. The most basic use is first-order logic extended with additional "theories" to reason about relevant topics such as integer/real arithmetic, motion (e.g., Newton's laws), spatial relationships, etc. The reasoning can be done by highly popular Satisfiability Modulo Theory (SMT) based tools such as Z3 [30] and YICES [12], which can routinely solve substantial practical problems. The three crucial parts in an SMT model are (a) Rules of Inference (RoIs), (b) Various feasibility constraints (e.g., walking speed <2 m/s), and (c) Groundings or facts determined from the environment (e.g., by analysis of objects in individual frames

[2]It is also possible to consider the strength of the assertion in the form of weight, but we have not done so in this paper for simplicity.

along with perspective correction). We can also define higher-level concepts as reusable functions suitable in the "logic program". For example, consider the function "following(V1, V2)" that asserts that car V1 is following car V2. This can be defined as the relative separation between V1 and V2 in a sequence of frames. It is even possible to establish the truth value of such a function directly via machine learning techniques.

Although SMT-based reasoning is adequate for this paper, explicit logic reasoning demonstrates its adaptability by handling more complex situations involving temporal ordering or real-time aspects. This adaptability is showcased through numerous extensions such as [4], [7], [9], [17], [40] as used in [32], [33].

Note that explicit logic reasoning differs from the so-called neuro-symbolic AI techniques [10], [22], which typically embed constraints into the loss function and then train the model. There is no loss function or training involved in explicit logic reasoning. Although we do not use it in this paper, it is possible to introduce fuzziness and hard/soft assertions in such reasoning [32].

C. Fine Tuning Algorithm

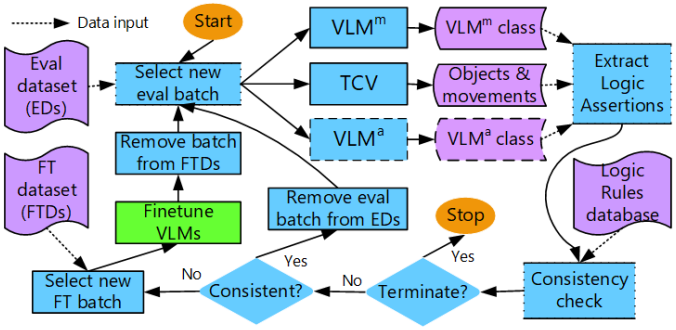

Fig. 2: Application Directed Fine-tuning of VLMs

With all the pieces in place, we describe the overall fine-tuning algorithm as illustrated in Fig. 2. The flowchart and description here are somewhat simplified for clarity, and variations may be used in an actual implementation. The purple boxes show the input/output data and the blue/green boxes show operations. We start with two datasets of input images (or videos): the fine-tuning dataset (FTD) and the evaluation dataset (ED). As stated earlier, EDs need not be labeled, although they are labeled in our case and allow the study of both consistency-based and accuracy-based evaluations. Ideally, the eval dataset should include one or more inputs from each VLM^m class, whereas each fine-tuning batch focuses on a specific class(es) of VLM^m that fails the consistency checks. *Note that the eval dataset differs from the test dataset, which will be used to test the performance after fine-tuning.*

The shown "Start" of the fine-tuning loop uses a batch of inputs from ED, depicted as *eval-batch* in Fig. 2, for each evaluation. It runs the eval-batch through VLM^m, VLM^a (if used), and TCV inferencing steps and collects the results. The outputs provide the "groundings" for the relevant assertions in the logic representation of the detected classes and proxy

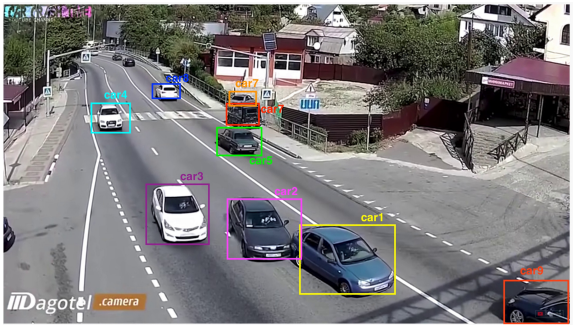

Fig. 3: Finding key objects in the scene

activities and enable consistency checks with the help of the Logic rules database, which is prebuilt. For example, in Fig. 3, the TCV (YoLov8) gives the bounding boxes of all the cars, and we can assign them some pseudo-IDs, such as car1, car2, etc. The recognized class here by VLM^m would be class 1 (from the left column in Table IV), and by VLM^a will also be class 1 (from a right column in Table IV). The proxy activity set $\mathbb{S}_1^m(A^p)$ includes the activities (#1,#3) stated abstractly in Table VII. These would be grounded in the analysis of the shown frame and a few prior frames in the video. Thus, the grounding of #1 ("A car behind another car in the same lane") will require the pseudo-IDs of the two vehicles and the truth value of the statement. Thus, after grounding, we obtain a statement like: "car2 moving behind car1 in the same lane". This grounding will be done in the logic domain (not natural language), and it will mark the corresponding assertion as true.

Following the grounding of all the assertions, we use SMT to check the consistency between the VLM outputs and grounded assertions. Suppose no inconsistency is observed for an eval-batch. In that case, we remove it from ED.[3] In case of inconsistency, the SMT framework provides the offending assertions (i.e., the subset of assertions that failed). This provides us with the VLM class for which more fine-tuning is needed. Thus, the next step is to choose a batch from FTD, depicted as *FT-batch* in Fig. 2. Subsequently, this batch is removed from FTDs, and the evaluation resumes.

To avoid clutter, the figure shows a single lumped termination condition – these checks would be placed in suitable places in the real code. The termination occurs if we run out of eval batches (i.e., only the just processed one is remaining), run out of fine-tuning batches, exceed a fine-tuning time limit, or the consistency measure (computed over a few iterations) stops improving.

Initially, both VLMs are fine-tuned using a set of randomly selected labeled inputs. In the case of video-based VLMs, we chop longer videos into small ones so that each video focuses on only one class of interactions as far as possible.[4] We label (or caption) these video segments according to the

[3]This action rests on the assumption of monotonicity, i.e., after an evaluation batch passes all the tests, it should not fail in future iterations, and thus is no longer helpful to expose weakness in VLM training. Non-monotonicity is possible but was not observed.

[4]Even with very short video segments, it is possible to have more than one activity. Such situations can be recognized by defining additional composite classes for conditions that are likely to occur together sufficiently often. It is expected that the number of such combinations will be small.

requirements of the specific VLM used. For image-based VLMs, we label each frame in the video segment identically.

IV. Details of Experimental Setup

A. VLMs Used For Evaluation

We have evaluated our finetuning methodology on image- and video-based VLMs [50]. We also consider both LLM-backed VLMs and those that merely do image-text matching. Each of these uses a different strategy to align vision and language models. By analyzing these diverse models, we aim to derive conclusions with a certain degree of generality.

MiniGPT-4 uses BLIP-2 (Bootstrapping Language-Image Pre-training) [23], which defines two trainable layers to align a frozen vision transformer model with a frozen LLM model. MiniGPT4-Video [5] is an extension of MiniGPT-4 that processes a sequence of frames and not only considers visual content but also incorporates textual conversations, allowing the model to answer queries involving both visual and text components effectively. X-CLIP [27] is a popular video-text matching-based VLMs. X-CLIP is designed for video-text retrieval and generates multi-grained visual and textual representations. Video-LLaMA [49] is a state-of-the-art multi-modal framework that allows LLMs to comprehend audio and video signals.

In recent years, the selective state space model (SSM) has emerged as an appealing alternative to transformer-based VLMs [42]. For example, VideoMamba [24] is a purely SSM-based model tailored for video understanding. In a classic ViT style, VideoMamba seamlessly integrates the advantages of convolution and attention. It provides a linear-complexity method, which is attractive from a real-time situational understanding perspective. We have explored our finetuning mechanism using VideoMamba and found similar results as stated below.

B. Description of Datasets Used

We evaluated our fine-tuning methodology using three very different datasets. The first two involve limited activities, whereas the third one is much more open-ended. Our first dataset, called TU_DAT [20], concerns road traffic and contains diverse accident types, weather conditions, and videos collected in challenging environments. Fig. 3 shows a scene from this dataset showing a rear-end accident.

Our second dataset, the Taekwondo dataset, captures movements performed by Taekwondo athletes. Understanding the movement patterns is a crucial component of Taekwondo training, as explained in the Taekwondo America student manual [2]. This dataset has 35 videos, which feature either a single student or multiple students performing the movements in sequence for each belt pattern. Fig. 4 (a) shows the walking stance low block, and (b) shows the walking stance reverse punch of a student in a dark green belt pattern.

Fig. 4: Green belt movement patterns in Taekwondo

(a) (b)

Fig. 5: Few activities from Kinetics dataset

Our third dataset is the Kinetics-100 [16], a large video dataset focused on human actions. The list of action classes includes single-person actions (e.g., drawing, drinking), person-person actions (e.g., hugging, shaking hands), and person-object actions (e.g., opening gifts, mowing lawn, washing dishes). Kinetics-100 has 100 human action classes, with 400–1150 clips for each action, and each clip lasts around 10 seconds. Fig. 5 shows some activities from the Kinetics-100 dataset, such as (a) Lawn mowing and (b) Washing dishes.

Although we chose a rather large number of activities from this dataset for evaluation purposes, the choice would center on unusual or anomalous activities.

Table III shows the details of all three datasets considered in this paper. We have used several augmentation methods to acquire enough data volumes for both TU_DAT and Taekwondo datasets, using Keras built-in capabilities. Keras provides various methods for real-time image augmentation, where the enhancement is performed while the network processes each image. The employed augmentations encompass flipping, translation, shear, and rotation.

TABLE III: Dataset details

Dataset type	fine-tuning Data	Eval Data	Test Data
TU_DAT	200	75	45
Taekwondo	105	65	30
Kinetics	2000	800	350

C. Classes Used to Fine-Tune VLMs

The TU_DAT dataset contains several accident scenarios in road traffic, forming the classes for fine-tuning a VLM. Since our proposed method includes fine-tuning two VLMs, the videos in the TU_DAT dataset have been categorized into modeling accident scenarios for VLM^m and recognizing the relative position/movements of vehicles for VLM^a. The description of classes used in fine-tuning VLM^m and VLM^a on TU_DAT are shown in Table IV. The table shows classes with the same number side by side for VLM^m and VLM^a, reflecting a one-to-one relationship between them. This relationship will be further captured through the logic assertions used to check consistency.

For the taekwondo dataset, VLM^m is fine-tuned to recognize the leg movements of the students, while VLM^a is fine-tuned to identify the students' arm movements. The description of classes used in fine-tuning VLM^m and VLM^a on the Taekwondo dataset are shown in Table VI.

Both datasets exhibit simple activities that TCV can effectively identify but would need specialized training for the

TABLE IV: Description of Classes used for TU_DAT Dataset

No.	Classes in VLM^m	Classes in VLM^a
1	Car hit by another from behind	car moving in same direction and one behind another
2	Car hit by another car from side	Car moving in opposite direction and perpendicukar to each other
3	Car hit by another car from front	Car moving in opposite direction
4	Car hits a static object	Car moving very close to static object
5	Motorcycle hits a pedestrian	Motorcycle moving very close to in same or opposite direction or perpendicular to walking pedestrian
6	Traffic videos	Cars moving next to, across, one behind another, in same or opposite direction
7	Not defined	Car & motorcycle moving next to one another
8	Not defined	Pedestrians walking

TABLE V: Description of Classes Used for Kinetics Dataset

No.	VLM^m Classes	VLM^a Classes
1	Arm wrestling	Using one arm in gripping the opponent's hand, applying pressure, stabilizing the body with both the legs, adjusting posture for leverage
2	Baking cookies	Mixing ingredients, rolling out dough, shaping cookies with hands, using legs to reach the oven, and opening it to place the baking tray inside
3	Brushing teeth	Using one hand to hold the toothbrush, apply toothpaste with other hand, and move it in circular motions to clean the teeth and gums
4	Cart wheeling	Both hands push off the ground and support the body while the legs kick up and rotate in the air to complete the cartwheel
5	Cheer leading	Using both hands to perform motions like clapping, waving, or holding pom-poms, while legs execute jumps, kicks, and stunts to enhance cheer routines
6	Moving Lawn	Using one of the hands to start the mower, adjust steering and cutting height, using both legs in pushing or guiding the mower and walk around obstacles
7	Washing dishes	Using both hands to scrape the dishes, applying soap to the dishes and rinsing the dishes with water

TABLE VI: Classes Used for Taekwondo Dataset

No.	Classes in VLM^m	Classes in VLM^a
1	Left leg still, right leg still	Left arms and right arms out
2	Left leg still, right leg forwards	Left arms out, right arms folded
3	Left leg still, Right leg backwards	Left arms folded, right arms out
4	Right leg still, Left leg forwards	Left arms and right arms folded
5	Right leg still, Left leg backwards	Left arms on the head, right arms folded
6	Left leg forward, Right leg backwards	Right arms on the head, left arms folded
7	Right leg forward, Left leg backwards	Not defined

kinetics dataset, making VLM^a attractive. We have carefully chosen approximately 50 pertinent classes (activities) out of 100 activities.[5] The comprehensive description of all 50 activities has been omitted due to space constraints; therefore, we present the descriptions of some of the selected activity classes utilized in fine-tuning VLM^m and VLM^a on Kinetics in Table V.

D. Identification of Proxy Activities

To illustrate the derivation of proxy activities, we start with a sample enumeration of simple activities, which in the traffic context are simply the spatial relationships between the key objects. These are listed informally in the top part of Table VII but can be easily expressed as logic assertions. Note that we have expressed the relationships generically since we want to be able to check them for any pair of cars or a car and a pedestrian. The object IDs will come from processing actual videos, as described next.

TABLE VII: Illustrating proxy activity identification

Possible spatial relationships (informal)
A car behind another car in the same lane
A car facing another car in the same lane
A car moving closer to next car
A car moving into next lane in same direction
A car moving into next lane in opposite direction
A pedestrian in a traffic lane.
A car moving closer to a pedestrian
Grounding of spatial relationships (informal)
car1 and car2 moving one behind another
car1, car2, car3 & car4 traveling in same lane
car1 is following car2 at very close distance
car1 and car9 traveling in the opposite lanes
car7 is parked and not moving
car5, car8, & car9 traveling in the same lane

TABLE VIII: Assertions for Reasoning

Variables: car1, car2.., car9 are integers
Functions: Boolean, each with one Integer argument
move_behind(), move_very_close(), move_opp_dirn()
move_same_dirn() car_hit_from_behind()
Groundings:
move_behind(car1,car2) $\wedge$ move_very_close(car1,car2) $\wedge$
move_constant_dirn(car1) $\wedge$ move_constant_dirn(car2) $\wedge$
move_constant_dirn(car5) $\wedge$ move_constant_dirn(car8) $\wedge$
move_constant_dirn(car8) $\wedge$ move_constant_dirn(car9) $\wedge$
move_opp_dirn(car1,car9) $\wedge$ move_behind(car1,car2) $\wedge$
move_very_close(car1,car2) $\Longrightarrow$ car_hit_from_behind (car1,car2)

We use Yolov8 to identify the key objects in the input images and track the objects across frames (in the case of videos) to "ground" the situation in terms of objects, pseudo-object IDs, positions, distances, and relative movements. For example, for the scene in Fig. 3, the identified (grounded) relationships are listed in the bottom part of Table VII informally. Again, for actual processing, we need to turn these into proper logic assertions, shown in Table VIII.

Given the groundings, we can determine which assertions in the top part of Table VII hold and for which cars. This represents a step in the process we described abstractly in section III-A to automatically determine proxy activities corresponding to each class identified by the VLM. The class is about a car being hit by another car from behind.

[5]Selecting 50 out of 100 is done for stress testing purposes; typically, only a small percentage of activities would be worth monitoring.

E. Fine-Tuning Specifics

We start by illustrating the mechanics of our fine-tuning with an example shown in Table IX. We use a rear-end accident scenario from the TU_DAT dataset using XCLIP VLM. After the initial fine-tuning stage, VLM^m and VLM^a yield the captions as shown in the top two lines in the panel below. The two outputs are inconsistent, which is affirmed by not having the expected class correspondence between VLM^m and VLM^a. Therefore, we select and retrieve the videos with the labels as depicted for additional fine-tuning of both VLMs.

TABLE IX: Illustration of Consistency Check Based Fine Tuning

VLM^m Output: Matching Caption: car hit by another from behind
VLM^a Output: Matching Caption: car moving in opposite direction

Consistency Check: NO - VLM^m output is inconsistent with VLM^a output
Retrieve Videos with labels: carhit by another from behind, car moving in same directions and one behind another
Fine Tune VLM^m and VLM^a
Select a new eval batch: To determine consistency between VLM^m and VLM^a outputs
Consistency Check: YES - VLM^m output is consistent with VLM^a output

For our results, we chose a fixed fine-tuning batch size of 20 videos. This is somewhat arbitrary, and one could vary the number as well. To make a fair comparison, we execute the loop four times for both directed and undirected cases, each using 20 videos. We stopped at four iterations since the improvement in consistency appeared to be stalled after that. All experiments were performed on a server with two NVIDIA RTX A6000 GPUs, each equipped with 10752 CUDA cores and 48GB GDRR6 memory. Our results *do not* assume parallel inferencing (see section III-C).

V. Experimental Results

The proposed fine-tuning framework requires evaluation in three critical areas: accuracy, consistency, and overhead. Accuracy is defined as the fraction of test cases whose classification matches the ground truth. For consistency, we introduce a measure called *Consistency Improvement Factor* (CIF), which is computed during the fine-tuning procedure. We define CIF as $(n_b - n_e)/n_b$ where n_b is the number of inconsistencies recorded before the fine-tuning, and n_e is the number of inconsistencies at the conclusion of the fine-tuning procedure. We compare the accuracy and CIF for directed vs. undirected fine-tuning methods. To ensure fairness, for the datasets with fewer main/auxiliary activities (TU_DAT and Taekwondo), we conducted both directed and undirected fine-tuning for an equivalent number of iterations. For the Kinetics dataset, which comprises 50 activities, we have performed both directed and undirected fine-tuning for the same duration. The overhead issue concerns fine-tuning time, inference time, and justification time.

The results in the following show that the directed method consistently outperforms the undirected one in all cases. Furthermore, this differential applies with both image-based VLMs, such as Minigpt4 [53] and its video-based version, MiniGPT4-video [5]. We have also demonstrated our fine-tuning mechanism on transformer-based VLMs that includes an LLM as the backend like Video-Llama [49], and more recent non-transformer, state space model(SSM) based VLMs like VideoMamba [24]. The same applies to models not backed by LLMs, such as XClip [27] and Video-MAE [41]. We also show that the improvement is sustained for two datasets, one relating to road traffic and traffic accidents and the other to the Taekwondo classroom.

A. Achieved Classification Accuracy

The most important result for our (consistency-driven) fine-tuning approach is the classification accuracy achieved on the test-dataset. Table X shows this for all three datasets and for both VLM^m and VLM^a. It is seen that the directed fine-tuning significantly outperforms the undirected fine-tuning in *all* cases and for both VLMs. However, the results depend on the VLM and the dataset.

TABLE X: Accuracy of fine-tuning (Consistency-Driven)

VLMs	Datasets	Undirected		Directed	
		VLM^m	VLM^a	VLM^m	VLM^a
MiniGPT4	TU_DAT	73.14	72.5	82.14	82.5
	Taekwondo	72.1	71.7	81.1	81.45
	Kinetics	79.6	78.4	84.15	84.78
MiniGPT4-V	TU_DAT	75.4	74.15	83.3	83.55
	Taekwondo	73.41	73.75	82.81	82.6
	Kinetics	80.15	79.85	84.4	84.2
Video-LLaMa	TU_DAT	78.25	78.35	85.12	85.84
	Taekwondo	77.5	77.85	85.22	85.1
	Kinetics	82.15	82.85	88.54	88.25
Video-Mamba	TU_DAT	76.6	75.55	81.3	81.42
	Taekwondo	76.41	76.8	80.85	80.8
	Kinetics	80.15	79.35	83.85	83.14

Next, we compare our consistency-driven fine-tuning approach against the accuracy-driven approach on the test data. Recall that the accuracy-driven fine-tuning selects inputs based on the observed inaccuracies on eval-data (see section II-C). The accuracy-driven fine-tuning does not require any TCV or VLM^a, but on the downside, it needs labeled eval-data and cannot provide justifiability during inference time. Table XI shows the accuracy-driven results for all three datasets. It is seen that the accuracy here is almost the same as in Table X; *; thus, we are not losing anything by following the consistency-driven approach.*[6]

TABLE XI: Accuracy of fine-tuning (Accuracy-Driven)

VLMs	Datasets	Undirected		Directed	
		VLM^m	VLM^a	VLM^m	VLM^a
MiniGPT4-V	TU_DAT	74.5	75.5	82.0	82.52
	Taekwondo	72.8	72.75	82.1	81.5
	Kinetics	81.25	79.5	85.0	85.25
Video-Mamba	TU_DAT	75.8	76.25	80.25	80.0
	Taekwondo	77.15	75.5	80.12	80.35
	Kinetics	79.4	80.55	84.6	84.1

B. CIF Results on TU_DAT/Taekwondo Datasets

Table XII shows the achieved CIF for TU_DAT and Taekwondo dataset using X-CLIP, Video-MAE and MiniGPT4

[6]The undirected results are slightly different than in Table X, possibly due to randomness; they should ideally be the same.

(image-based), MiniGPT4-Video, Video-Llama and Video-Mamba respectively.

It is clear that our directed fine-tuning surpasses undirected fine-tuning in all cases by a very significant margin. Note that the substantial improvement in consistency persists for two very different types of videos (road traffic vs. taekwondo), confirming that the improvement is not tied to the video characteristics.

TABLE XII: CIF results on TU_DAT and Taekwondo datasets

VLM Models	Task Datasets	Undirected		Directed	
		VLM^m	VLM^a	VLM^m	VLM^a
X-CLIP	TU_Dat	54.5	55.15	74.25	73.65
X-CLIP	Taekwondo	48.71	48.15	69.85	70.05
VideoMAE	TU_Dat	52.04	52.41	72.65	73.25
VideoMAE	Taekwondo	48.5	49.06	69.8	69.31
MiniGPT4	TU_DAT	59.78	60.41	75.51	74.35
MiniGPT4	Taekwondo	59.78	60.41	75.71	75.41
MiniGPT4-Video	TU_Dat	71.45	71.8	86.35	85.125
MiniGPT4-Video	Taekwondo	68.40	68.1	83.10	83.08
Video-Llama	TU_Dat	72.16	72.41	86.85	87.32
Video-Llama	Taekwondo	42.57	42.05	84.60	85.08
VideoMamba	TU_DAT	61.95	61.41	80.85	80.4
VideoMamba	Taekwndo	61.60	61.75	76.52	75.45

We also examine the impact of preceding the use of VLM^a. This is possible because of the simplicity of VLM^a tasks in these two datasets. Table XIII shows the *accuracy* of VLM^m on test data with and without using VLM^a. In the latter case, the consistency check is done between the proxy activities tied to VLM^m and those tied to VLM^a. We used MiniGPT4-Video as the main VLM in this case. Although the directed method still provides a substantial jump in accuracy, the result is lower by a few percentage points without using VLM^a.

C. CIF Results on Kinetics Dataset

To assess the significance of incorporating VLM^a in datasets that involve hundreds of complex main activities and multiple proxy activities, we investigated the effectiveness of our fine-tuning approach on the Kinetics dataset. In this dataset, the classes in VLM^m correspond to the activity classes included in the dataset. In contrast, the classes in VLM^a correspond to the more straightforward activities associated with each activity class. For instance, the activity class "shaking hands" in VLM^m is represented by two individuals shaking hands. In comparison, classes in VLM^a include descriptions such as "two individuals standing straight up, facing each other, and extending their right hands." Initially, both VLMs are fine-tuned on a random set of videos, followed by the selection of fine-tuning videos belonging to the classes that show inconsistency.

TABLE XIII: Fine-tuning accuracy w/ & w/o VLM^a

Case	TU_DAT	Taek.
Undirected	75.4	73.41
With VLM^a	83.3	82.81
W/o VLM^a	81.0	80.65

We follow the same fine-tuning procedure from Fig. 2 using a batch size of 50. Table XIV compares the CIF measure for MiniGPT4, MiniGPT4-Video, and Videomamba. (Others are similar and not reported.) Note that we are running directed and undirected fine-tuning for an equal amount of total time. Consequently, a significant portion of the total time is consumed in the fine-tuning of VLM^a. Despite this, the CIF of the directed approach is higher by about ten percentage points. This robust validation of our earlier claim that the use of VLM^a can be easily justified in large, complex datasets should reassure about the soundness of our conclusions.

TABLE XIV: CIF Results on Kinetics Dataset

VLMs	Undirected	Directed
MiniGPT4	74.25%	83.7 %

D. Analysis of Fine-Tuning Time

We also compare the time required for both directed and undirected fine-tuning approaches for all three VLMs under consideration. As stated earlier, we use four iterations of fine-tuning for both directed and undirected cases, each using 20 videos. The time spent on each iteration consists of two parts, reported as average per epoch, over 500 epochs: (a) actual fine-tuning time and (b) preparation time. For the undirected case, prep-time randomly retrieves 20 videos from the disk. For directed cases, prep-time *also* includes the overhead of running YOLOv8, querying VLM^m and VLM^a, generating assertions, and using them for consistency checking.

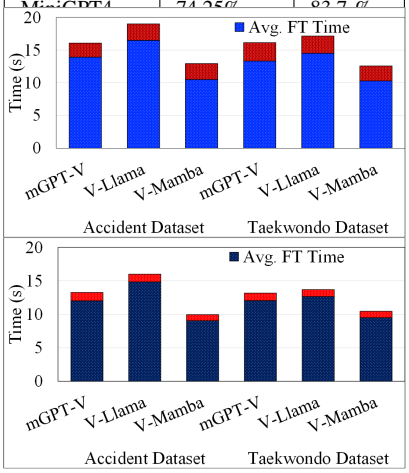

Fig. 6: Per-epoch prep and fine-tuning Time for (a) Directed & (b) Undirected cases

Figs. 6 (a) and (b) show the average per-epoch fine-tuning time and fine-tuning prep-time, respectively, for both undirected and directed cases. As expected, the fine-tuning time is almost identical in both cases and is in the ∼10-12 sec range. The prep time is much shorter; a significant piece is retrieving and loading videos from the disk. The time taken by other pieces of directed fine-tuning is relatively modest.

E. Analysis of Justification Time

To evaluate justifiability, we take out the fine-tuning section during inferencing (thus breaking the loop) but retain other parts. In this case, each inference will also be accompanied by the following information: (1) Output justified by alluding to the consistency between VLMs and across VLMs and TCV-based proxy activity detection, and (2) Output marked as faulty along with a reason why it is considered questionable.

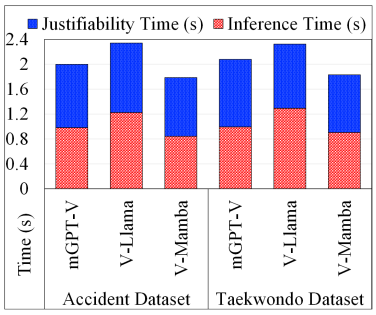

Fig. 7: Justification/Inference Time

Fig. 7 displays the inference and justifiability time for MiniGPT4-Video, Video-Llama, and Video-Mamba on both datasets. Note that the justifiability time differs from the fine-tuning prep time reported above since we no longer have the significant overhead of retrieving videos from disk for fine-tuning.

It is seen that the justifiability time is about the same as the inference time. This should be reasonable for critical applications where inaccuracy can have serious consequences. For other applications, we run justifiability less often. One way to do this is to do the justification periodically. Another way is to observe which classes have reliability issues and then use a sequential procedure where VLM^m first predicts the tentative class, and then VLM^a is run if the identified VLM^m class is among the less reliable ones.

F. Catastrophic Forgetting in VLMs

VLM/LLM progress continues to follow the path of an ever more significant number of parameters (or weights) whose values depend in unknown ways on the vast amount of disparate pretraining data. When these models are fine-tuned for a specific purpose, some weights are updated to enhance performance on the targeted activities. But, in the process, this may severely degrade the performance of other related activities. This catastrophic forgetting (CF) phenomenon is well-known but poorly understood [18], [26], [48]. However, its presence means that it may not be sensible to train the same VLM with both main and auxiliary activities.

TABLE XV: Catastrophic forgetting

Ver	Fine-Tuning target	Accu.
V1	None (Orig. VLM)	36%
V2	V1 on VLM^m classes	44%
V3	V2 on VLM^a classes	32%

Table. XV illustrates this in the context of XCLIP VLM on the accident dataset. It shows the CIF of 3 versions of the VLM. The first one is without any fine-tuning (denoted as version V1). Version V2 results from fine-tuning V1 to do VLM^m classification, and version V3 results from fine-tuning V2 for VLM^a classes. It is seen that V3's CIF is lower than V1's, thereby showing the CF phenomenon. This experiment demonstrates why we kept VLM^m and VLM^a separate in our fine-tuning methodology.

VI. Related Work

Numerous VLMs exist and continue to emerge rapidly. Some of these work only with images (e.g., MiniGPT4 [53], LLAVA [25], Clip [38]), while others work with (short) videos (e.g., Video-LLAMA [49], Video-ChatGPT [28], X-Clip [27], MiniGPT4-video [5], VideoMamba [24]).

The logical reasoning used in this paper involves the extension of first-order deductive reasoning using explicit Rules of Inference (RoIs). This differs from other recent notions of "reasoning" using VLM/LLM outputs [8], [31], [45], [46]. Such reasoning fine-tunes the VLM/LLM to ask questions and generate answers directly, potentially in conjunction with external information obtained via web searches. The claims of reasoning ability of LLMs have been surveyed and questioned in [15], [43].

In the space of TCV and, more generally, deep learning, the issue of reasoning is often described as neuro-symbolic AI [10], [14], [22], [39]; however, it is mainly in the form of indirectly enforcing the constraints in neural net operations or loss function. For example, the popular Logic Tensor Networks (LTN) [6] enforces logic constraints implicitly and approximately by using differentiable extensions of Boolean operations [19]) to avoid the problem of exploding or vanishing gradients. Explicit logic reasoning approaches are relatively sparsely explored [3], [37]. Ref [44] attempts to generate captions for videos based on captions of individual images by teaching it to recognize the temporal context. Ref [32], [33] attempts to use explicit reasoning for accident and driver behavior characterization.

A conventional fine-tuning approach adds a classifier on top of the visual backbone [36] or additional feature adapter [13]. Another type of fine-tuning is referred to as prompt-based, where the classification weights are synthesized from natural language describing the classes of interest and which involves fine-tuning the prompt to maximize the ground truth token [51], [52]. The authors in [47] propose an algorithmic framework that utilizes Reinforcement Learning to fine-tune VLMs directly. They use VLMs' chain of Thought (CoT) reasoning capability to fine-tune them. However, the main drawback of such a framework is that CoT reasoning plays a critical role in enhancing the performance of VLMs during fine-tuning.

VII. Conclusions

In this paper, we propose a novel consistency-driven fine-tuning approach for VLMs, which combines traditional computer vision (TCV) to recognize details with explicit logical reasoning to improve VLMs' performance. The mechanism can substantially reduce the labeled data needs of fine-tuning and achieve considerably higher accuracy than a mechanism that does not choose the input intelligently. It also provides a justification mechanism that can continue to be used at inference time and a vital sanity check mechanism for situational awareness applications. The proposed mechanism is quite general in that we can decide the level of complexity that we wish to introduce in consistency checking. We illustrated this tradeoff by checking consistency with and without the auxiliary VLM, but its more thorough investigation remains an area for future work.

References

[1] Ábrahám, E., Kremer, G.: Satisfiability checking: Theory and applications. In: Software Engineering and Formal Methods: 14th International Conference, SEFM 2016, Held as Part of STAF 2016, Vienna, Austria, July 4-8, 2016, Proceedings 14. pp. 9–23. Springer (2016) 3

[2] America, T.: Taekwondo student manual. https://taekwondoamerica.org/wp-content/uploads/2017/09/Student-Manual-2012.pdf (2011) 5

[3] Artikis, A., et al.: A logic programming approach to activity recognition. In: Proceedings of the 2nd ACM international workshop on Events in multimedia. pp. 3–8 (2010) 9

[4] Artikis, A., et al.: An event calculus for event recognition. IEEE TKDE **27**(4), 895–908 (2014) 4

[5] Ataallah, K., Shen, X., Abdelrahman, E., Sleiman, E., Zhu, D., Ding, J., Elhoseiny, M.: Minigpt4-video: Advancing multimodal llms for video understanding with interleaved visual-textual tokens. arXiv preprint arXiv:2404.03413 (2024) 5, 7, 9

[6] Badreddine, S., Garcez, A.d., Serafini, L., Spranger, M.: Logic tensor networks (ltn). Artificial Intelligence **303**, 103649 (2022) 9

[7] Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., Tonetta, S.: The nuxmv symbolic model checker. In: CAV. pp. 334–342 (2014) 4

[8] Chen, Z., Zhou, Q., Shen, Y., Hong, Y., Zhang, H., Gan, C.: See, think, confirm: Interactive prompting between vision and language models for knowledge-based visual reasoning. arXiv preprint arXiv:2301.05226 (2023) 9

[9] Chisalita, I., et al.: Traffic accidents modeling and analysis using temporal reasoning. In: ITSC. pp. 378–383 (2004). https://doi.org/10.1109/ITSC.2004.1398928 4

[10] De Raedt, L., Dumančić, S., Manhaeve, R., Marra, G.: From statistical relational to neuro-symbolic artificial intelligence. arXiv preprint arXiv:2003.08316 (2020) 4, 9

[11] Dou, Z.Y., Xu, Y., Gan, Z., Wang, J., Wang, S., Wang, L., Zhu, C., Zhang, P., Yuan, L., Peng, N., et al.: An empirical study of training end-to-end vision-and-language transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18166–18176 (2022) 1

[12] Dutertre, B.: Yices 2.2. Computer-Aided Verification (CAV'2014) **8559**, 737–744 (July 2014) 3

[13] Gao, P., Geng, S., Zhang, R., Ma, T., Fang, R., Zhang, Y., Li, H., Qiao, Y.: Clip-adapter: Better vision-language models with feature adapters. International Journal of Computer Vision **132**(2), 581–595 (2024) 9

[14] Garcez, A.d., Lamb, L.C.: Neurosymbolic ai: the 3rd wave. arXiv preprint arXiv:2012.05876 (2020) 9

[15] Huang, J., Chang, K.C.C.: Towards reasoning in large language models: A survey. arXiv preprint arXiv:2212.10403 (2022) 9

[16] Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al.: The kinetics human action video dataset. arXiv preprint arXiv:1705.06950 (2017) 5

[17] Konur, S.: A survey on temporal logics for specifying and verifying real-time systems. Frontiers of Computer Science **7**(3), 370–403 (2013) 4

[18] Kotha, S., Springer, J.M., Raghunathan, A.: Understanding catastrophic forgetting in language models via implicit inference. arXiv preprint arXiv:2309.10105 (2023) 9

[19] van Krieken, E., Acar, E., van Harmelen, F.: Analyzing differentiable fuzzy logic operators. Artificial Intelligence **302**, 103602 (2022) 9

[20] Kumar., P.P.: Tu-dat dataset. https://github.com/pavana27/TU-DAT (2021) 5

[21] Lampert, C.H., Nickisch, H., Harmeling, S.: Attribute-based classification for zero-shot visual object categorization. IEEE transactions on pattern analysis and machine intelligence **36**(3), 453–465 (2013) 2

[22] Lee, J.H., Sioutis, M., Ahrens, K., Alirezaie, M., Kerzel, M., Wermter, S.: Neuro-symbolic spatio-temporal reasoning. arXiv preprint arXiv:2211.15566 (2022) 4, 9

[23] Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In: International conference on machine learning. pp. 19730–19742. PMLR (2023) 5

[24] Li, K., Li, X., Wang, Y., He, Y., Wang, Y., Wang, L., Qiao, Y.: Videomamba: State space model for efficient video understanding. arXiv preprint arXiv:2403.06977 (2024) 5, 7, 9

[25] Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. arXiv preprint arXiv:2304.08485 (2023) 9

[26] Luo, Y., Yang, Z., Meng, F., Li, Y., Zhou, J., Zhang, Y.: An empirical study of catastrophic forgetting in large language models during continual fine-tuning. arXiv preprint arXiv:2308.08747 (2023) 9

[27] Ma, Y., Xu, G., Sun, X., Yan, M., Zhang, J., Ji, R.: X-clip: End-to-end multi-grained contrastive learning for video-text retrieval. In: Proceedings of the 30th ACM International Conference on Multimedia. pp. 638–647 (2022) 5, 7, 9

[28] Maaz, M., Rasheed, H., Khan, S., Khan, F.S.: Video-chatgpt: Towards detailed video understanding via large vision and language models. arXiv preprint arXiv:2306.05424 (2023) 9

[29] Maaz, M., Rasheed, H., Khan, S., Khan, F.S.: Video-chatgpt: Towards detailed video understanding via large vision and language models. arXiv:2306.05424 (2024) 3

[30] de Moura, L., Bjørner, N.: Z3: An efficient smt solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS. pp. 337–340 (2008) 3

[31] Paranjape, B., Lundberg, S., Singh, S., Hajishirzi, H., Zettlemoyer, L., Ribeiro, M.T.: Art: Automatic multi-step reasoning and tool-use for large language models. arXiv preprint arXiv:2303.09014 (2023) 9

[32] Pradeep, P., Kant, K., Pal, A.: C-far: A compositional framework for anomaly resolution in intelligent transportation system. IEEE Trans. on Intelligent Transportation Systems (Aug 2022). https://doi.org/10.1109/TITS.2022.3196548 3, 4, 9

[33] Pradeep, P., Kant, K., Pal, A.: Non-intrusive driver behavior characterization from road-side cameras. IEEE IoT Journal (Sept 2023). https://doi.org/10.1109/JIOT.2023.3285886 3, 4, 9

[34] Pradeep, P., Kant, K., Rienzo, F.D., Vallati, C., Pal, A.: Improper human posture recognition from rgb-d images. Submitted for publication, available at https://www.kkant.net/papers/posture_recognition_paper.pdf (Feb 2024) 3

[35] Pradeep, P., Pal, A., Kant, K.: Automating conflict detection and mitigation in large-scale iot systems. Proc. of CCGrid Conference (May 2021). https://doi.org/10.1109/CCGrid51090.2021.00063 3

[36] Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021) 9

[37] Ramirez-Amaro, K., Kim, E.S., Kim, J., Zhang, B.T., Beetz, M., Cheng, G.: Enhancing human action recognition through spatio-temporal feature learning and semantic rules. In: 2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids). pp. 456–461. IEEE (2013) 9

[38] Rasheed, H., Khattak, M.U., Maaz, M., Khan, S., Khan, F.S.: Fine-tuned clip models are efficient video learners. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6545–6554 (2023) 9

[39] Sarker, M.K., Zhou, L., Eberhart, A., Hitzler, P.: Neuro-symbolic artificial intelligence. AI Communications **34**(3), 197–209 (2021) 9

[40] Vlassopoulos, C., Artikis, A.: Towards a simple event calculus for runtime reasoning (rtec). In: COMMONSENSE (2017) 4

[41] Wang, L., Huang, B., Zhao, Z., Tong, Z., He, Y., Wang, Y., Wang, Y., Qiao, Y.: Videomae v2: Scaling video masked autoencoders with dual masking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14549–14560 (2023) 7

[42] Wang, X., Wang, S., Ding, Y., Li, Y., Wu, W., Rong, Y., Kong, W., Huang, J., Li, S., Yang, H., et al.: State space model for new-generation network alternative to transformers: A survey. arXiv preprint arXiv:2404.09516 (2024) 5

[43] Wang, Y., Chen, W., Han, X., Lin, X., Zhao, H., Liu, Y., Zhai, B., Yuan, J., You, Q., Yang, H.: Exploring the reasoning abilities of multimodal large language models (mllms): A comprehensive survey on emerging trends in multimodal reasoning. arXiv preprint arXiv:2401.06805 (2024) 9

[44] Wang, Z., Li, M., Xu, R., et al.: Language models with image descriptors are strong few-shot video-language learners. Advances in Neural Information Processing Systems **35**, 8483–8497 (2022) 9

[45] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems **35**, 24824–24837 (2022) 9

[46] Yang, Y., Zhang, X., Han, W.: Enhance reasoning ability of visual-language models via large language models. arXiv preprint arXiv:2305.13267 (2023) 9

[47] Zhai, Y., Bai, H., Lin, Z., Pan, J., Tong, S., Zhou, Y., Suhr, A., Xie, S., LeCun, Y., Ma, Y., et al.: Fine-tuning large vision-language models as decision-making agents via reinforcement learning. arXiv preprint arXiv:2405.10292 (2024) 9

[48] Zhai, Y., Tong, S., Li, X., Cai, M., Qu, Q., Lee, Y.J., Ma, Y.: Investigating the catastrophic forgetting in multimodal large language models. arXiv preprint arXiv:2309.10313 (2023) 9

[49] Zhang, H., Li, X., Bing, L.: Video-llama: An instruction-tuned audio-visual language model for video understanding. arXiv preprint arXiv:2306.02858 (2023) 5, 7, 9

[50] Zhang, J., Huang, J., Jin, S., Lu, S.: Vision-language models for vision tasks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence (2024) 1, 5

[51] Zhou, K., Yang, J., Loy, C.C., Liu, Z.: Conditional prompt learning for vision-language models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16816–16825 (2022) 9

[52] Zhou, K., Yang, J., Loy, C.C., Liu, Z.: Learning to prompt for vision-language models. International Journal of Computer Vision **130**(9), 2337–2348 (2022) 9

[53] Zhu, D., Chen, J., Shen, X., Li, X., Elhoseiny, M.: Minigpt-4: Enhancing vision-language understanding with advanced large language models. arXiv preprint arXiv:2304.10592 (2023) 7, 9