Reasoning over the Behaviour of Objects in Video-Clips for Adverb-Type Recognition

Amrit Diggavi Seshadri Imperial College London asd20@ic.ac.uk Alessandra Russo Imperial College London a.russo@imperial.ac.uk

Abstract

In this work, following the intuition that adverbs describing scene-sequences are best identified by reasoning over high-level concepts of object-behavior, we propose the design of a new framework that reasons over object-behaviours extracted from raw-video-clips to recognize the clip's corresponding adverb-types. Importantly, while previous works for general scene adverb-recognition assume knowledge of the clips underlying action-types, our method is directly applicable in the more general problem setting where the action-type of a video-clip is unknown. Specifically, we propose a novel pipeline that extracts human-interpretable object-behaviour-facts from raw video clips and propose novel symbolic and transformer based reasoning methods that operate over these extracted facts to identify adverb-types. Experiment results demonstrate that our proposed methods perform favourably against the previous state-of-the-art. Additionally, to support efforts in symbolic video-processing, we release two new datasets of object-behaviour-facts extracted from raw video clips - the MSR-VTT-ASP and ActivityNet-ASP datasets.

1 Introduction

In recent years, the task of recognizing the type of actions being performed in video-clips has gained much of the vision community's attention [3, 7, 9, 11, 18, 19, 20]. It is a relatively well studied problem, with practical applications in smart-home systems and robotics. Current state-of-the-art methods for action-type recognition fuse predictions from two-streams of convolutional neural networks (CNNs). One stream predicts action-type probabilities from stacked image-frames of the input video-clip while another stream predicts probabilities from stacked frames of the clip's optical-flow. The output from these two streams are fused together for inference. In particular, the Inflated 3D Convolutional Network (I3D) architecture [3] has demonstrated much success for action-type recognition by employing this two-stream paradigm with 3D-convolutional operations.

In contrast to action-type recognition - for which numerous architectures have been proposed, the problem of adverb-type recognition is less well explored. Adverbs further describe the nature and execution of generic action types, providing additional detail regarding intent, meaning and consequences. A device recording recipes in a kitchen for example might deem a cook in the action of "stirring" a pot **slowly** and **completely** to be performing a required and delicate step. The same action performed **fast** or **partially** on the other hand, might be of less consequence. Interestingly, adverbs can also prove useful even without any knowledge of the underlying action-type. We might for example deem all recordings that are performed **slowly** and **completely** to take precedence over **partially** executed work.

To our knowledge, there have been two architectures proposed to solve the task of adverbtype recognition in general-scene video clips [5, 6]. However, these previous methods both assume the availability of ground-truth action-types as a prerequisite for adverb-type recognition and follow the trend set by previous action-recognition systems - encoding video clips using an I3D backbone. These practices make them unsatisfactory for two important reasons.

Firstly, ground-truth action-types are not usually known for raw video clips, making the previous methods inapplicable in many scenarios. One might attempt to compensate for this by using a pretrained action-type predictor to feed into the adverb-recognition model. However, doing so is a non-trivial and complex task - as previous methods [5, 6] assume knowledge of over 100 distinct ground-truth action-type categories. Using predictions for more than 100 action-type categories invariably leads to incorrect and noisy input data - especially if the number of training samples are limited. Additionally, under such a bootstrapped framework, one would be forced to re-train both action-type and adverb-type predictors when new video-clips with new or unseen action-type emerge. Ideally, we would instead prefer to have a framework and model wherein adverb-type predictions are made without requiring any knowledge of the video-clip's action-types.

Secondly, while end-to-end black-box CNN models such as the I3D architecture have proved successful for action-type recognition, a key reason for this success has been the fact that CNNs excel at object recognition, and a video-clip's action-type is greatly constrained by the type of objects present within scenes. The same is not true for adverb-type recognition. Object-type may vary widely across different instances of the same adverb. A person cooking **slowly** for example presents a very different scene from a dog running **slowly** in a park. And while the use of optical-flow input does mitigate some of this problem by providing motion-related information, end-to-end CNN models fail to generalize well over such diverse scenes.

However, despite this added complexity of not being constrained by object-type, humans are usually able to easily identify adverb-types by reasoning over high-level concepts of object behaviour. Something happening **slowly** for example might be identified to mean that objects change very little between frames. Properties of other adverbs such as **partially** or **completely** are less straightforward to define, but again seem easier to identify by reasoning over higher-level concepts of object-behaviour than they are to identify by pattern-matching over diverse scenes that vary widely.

In this work, following the intuition that adverbs are best described by reasoning over higher-level concepts, we propose a novel framework that (1)Extracts discrete facts of object-behaviour from raw video clips (2)Reasons over those extracted facts to produce high-level summaries of object-behaviour (3)Predicts and aggregates adverb-types using down-stream models over those high-level summaries.

Importantly, unlike previous work for general scene adverb-recognition [5, 6], our framework does <u>not</u> assume any knowledge of the video-clip's action-type during training or inference making it directly applicable to the more general problem setting wherein the action-type of a video clip is unknown. Our main contributions are summarized as follows:

- We propose the design of a novel *action-free* framework for adverb-type recognition in video clips that extracts object-facts from raw video clips; reasons over those facts to learn high-level behaviour summaries; and makes predictions of adverb-types from those summaries.
- We propose a novel extraction phase for our framework that converts raw video clips to discrete Answer Set Programs (ASP) of facts capturing information regarding objects moving within each clip. Using this new extraction phase, we release two new datasets of object-behaviour-facts the 'MSR-VTT-ASP' and the 'ActivityNet-ASP' datasets.
- For the reasoning phase of our framework, we propose novel symbolic and transformerbased reasoning methods over our extracted ASP-facts to obtain higher-level summary vectors of object-behaviour.
- Finally, we evaluate the performances of the different symbolic and transformer-basedarchitectures that we propose within our framework, and make a comparison against the previous state-of-the-art.

Experiment results demonstrate that our new methods for adverb-type recognition perform favourably against the previous state of the art on video-clips from the MSR-VTT and ActivityNet datasets, providing a new means for adverb-type recognition when the action-type of a video-clip is unknown.

2 Related Work

Action-Type Recognition: Simonyan et al. [19] was first to propose a two-stream 2D CNN network for action-type recognition - that employs a separate stream to process image-frames and a separate stream to process their optical flow. This two-stream method outperformed the previous method of predicting actions from features pooled across video-frame snips [11] by a large margin. Subsequently, 3D CNNs [20] were shown to outperform their 2D counterparts by better preserving temporal information across input frame sequences, and building on these ideas, the Two-Stream Inflated 3D CNN (I3D) network was proposed [3] - using two streams of 3D convolutional networks over stacked frames of a video clip's image frames and optical flow. This I3D model significantly outperformed the previous methods and is employed as the backbone of a number of state-of-the-art action-type recognition systems [7, 9, 18]. However, as pointed out earlier, the two-stream and 3D CNN paradigms operate end-to-end, directly over raw pixel maps of image frames or optical flow and fail to cleanly separate out and reason over individual object-behaviours across time-steps.

Adverb-Type Recognition: Pang et al. [16] was first to explore the problem of adverbtype recognition in video clips, introducing the "Adverbs Describing Human Actions" (ADHA) dataset and employing a hybrid two-stream CNN along with expression detectors and human pose-estimates. However, their work addresses a problem setting different from the one that we are interested in. The ADHA dataset is focused on adverbs for human subjects, and places special focus on human pose and expression informed adverbs. We are interested in scenes comprising more general content that may not be human. Doughty et al. [5, 6] scaled up the problem of adverb-type recognition to general-scene video clips, and released adverb-annotations for subsets of video-clips from the HowTo100M[15], VATEX[21], MSR-VTT[23] and ActivityNet[2] datasets while proposing new architectures for the task. However, as mentioned earlier, both of these prior works [5, 6] assume knowledge of the video-clip's underlying action-types as a prerequisite for adverb-type recognition (with over 100 distinct ground-truth action-type categories), and they encode video-clips using an I3D backbone without attempting to reason over individual object-behaviours. In our work, we use the adverb-annotations released by Doughty et al. [6] for experiments on video-clips from the MSR-VTT and ActivityNet datasets - datasets for which raw video files are publicly available.

3 Method

Our adverb-type recognition framework (Figure 1a) comprises three phases - an Extraction phase, a Reasoning phase and a Prediction phase. The Extraction phase extracts separate discrete, and human-interpretable object-behaviour-facts for each object detected to be of interest within the video clip. The Reasoning Phase summarizes those facts across time-steps into summary vectors for each object. The Prediction Phase makes downstream classifications, using separate SVMs to classify between each adverb and it's antonym. Finally, as we obtain separate SVM predictions for each object detected to be of interest in a clip, we aggregate results by majority-voting.

3.1 Extraction Phase

Figure 6 shows a depiction of our extraction pipeline. Given a raw video clip, we first employ MaskRCNN [10] over delayed-captures of static frames from the video clip's image sequence - considering every fifth frame of the original clip. In doing so, we avoiding processing successive frames between which very little changes. MaskRCNN gives us a collection of predicted object-types and their corresponding bounding boxes with confidence scores between 0 and 1. We ignore all detections made with a confidence score less than 0.3, and flag all detected patches with low confidence scores between 0.3 and 0.5 as 'unknown' object-types¹. Patches detected with a confidence score above 0.5 are recorded along with their predicted object-types. To capture properties of motion for each of these detected object patches, we compute the pixel-wise Gunnar-Farneback optical flow [8] between consecutive delayed capture frames. These per-pixel optical-flow values are averaged within each detected object-bounding-box to give us a single average numeric value of optical-flow magnitude and a single average numeric value of optical-flow angle for each detected

¹To simplify our explorations and to reduce noise in the input data, in this work, we ignore 'unknown' type object-behaviours detected by our extraction phase - leaving reasoning over those less-confident object-facts as scope for future work.



(a) A depiction of our proposed framework for adverb-type recognition.

(b) A depiction of our video-to-objectbehaviour-facts (video-to-ASP) extraction phase.

Figure 1: A depiction of our proposed framework (a) and our video-to-object-behaviour-facts extraction phase (b).

object-patch. To filter these numerous detections, we then slide a non-overlapping sliding window over the delayed capture frames (each window detection corresponding to a single time-step), and we assume that (1)Of the objects detected in a frame, only objects moving faster than the frame's average are of interest for adverb-type recognition, (2)Of those filtered cases, only objects detected consistently in at least half the frames of the sliding window can be considered important enough for adverb-recognition. It is necessary for us to make these assumptions/choices to reduce the complexity of the problem faced by subsequent phases. Automatedly learning optimal property-extractions for adverb-type recognition is scope for future work and poses a significantly more challenging task.

A consistently detected object of interest has its properties averaged across a time-step's window, and these properties are recorded as Answer Set Programming (ASP) [14] facts as shown in Figure 6, where "detected(person, 2)" means that an object of type 'person' is detected at time-step 2. We also capture local temporal properties of objects such as 'operation-area' and 'movement-in-place' at each time step, and record the region of the frame 'cell_occupancy' that the object occupies for that time-step. To simplify processing, optical-flow angles are bucketed into discrete directions *north(n), north-east(ne), east(e),* etc.; while numeric-values besides optical-flow magnitude are thresholded to *very_small, small, medium, large* and *very_large*. The implementation details of these extracted predicate properties are discussed in greater detail in Appendix Section 6.1.

Our video-to-ASP extraction pipeline can be used to convert raw video-clips to ASP-programs in an online-fashion. However, to simplify our training and evaluation procedures and to further research efforts in symbolic and neuro-symbolic video-processing, we instead preprocess video-clips taken from the MSR-VTT and ActivityNet datasets using our proposed pipeline in an offline-manner and release new 'MSR-VTT-ASP' and 'ActivityNet-ASP' datasets consisting of extracted object-behaviour-facts and releavant background knowledge over predicate properties. Details of these new datasets are discussed further in Sections 4.

3.2 Reasoning Phase

3.2.1 Symbolic-Based Reasoning

In this work, we first consider employing the FastLAS inductive learning method [13] to automatedly learn indicator rules that plausibly define adverb-types. However, automatedly learning symbolic rules that reason over more than one time-step is an extremely challenging task (owing to the large number of possible variable groundings of rules). Rather than attempting to overcome the multi-time-step challenge, in this initial exploration, we consider learning a large number of simple single-time-step rules, that compositionally might inform overall adverb-type. In particular, for magnitude, angle, and operation_area predicates, we focus on learning range-rules that define upper or lower bounds of an object's predicate-properties at single time-steps, and for the *cell-occupancy* predicate, we focus on learning single-time step rules that outline rough left/right or up/down placement of an object within the frame. As an example, Figures 2a and 2b depict FastLAS learnt ASP range-rules that classify between categories of 'adverb_A' and 'antonym_A' motion for some collection of object-behaviour facts. According to these indicator rules, objects moving with an optical-flow magnitude between five and twenty at some time-step are considered to exhibit 'adverb_A' behavior, while those objects moving with optical-flow magnitude outside those limits are considered to exhibit 'antonym_A' behavior. These rules might not hold as universally true, however they are identified by FastLAS as plausibly explanations for some given batch of input behaviour-examples. Similarly, from the training

class(adverb_A,VO) :-	<pre>- magnitude(V0,V2,V1), less_than(V1,twenty_to_twfive,V_0_count), V_0_count <= 3, object(V0), speed(V1), time(V2).</pre>
class(antonym_A,VO)	<pre>:- magnitude(V0,V2,V1), less_than(V1,five_to_ten,V_0_count), object(V0), speed(V1), time(V2).</pre>
class(antonym_A,VO)	<pre>:- magnitude(V0,V2,V1), less_than(fifteen_to_twenty,V1,V_0_count), object(V0), speed(V1), time(V2).</pre>

(a) Batch-plausible ASP range-rules learnt to logically-explain (adverb_A, antonym_A) categories.

antonym_A adverb_A			A	antony	antonym_A							
												Т
	0	5	10	15	20	25	30	35	40	45	50	55

(b) A depiction of the ranges defined by the rules shown in Figure 2a.

Figure 2: A depiction of rules learnt by FastLAS for some batch of example object-behaviours labeled with (adverb_A, antonym_A) categories.

data, we learn indicator rules classifying between each (adverb, antonym) pair. To do so we sample small-balanced-batches of object-behaviour-facts from the training data - choosing 10 randomly sampled object-behaviours for each adverb, and 10 random sampled object-behaviours for it's antonym. We then run FastLAS separately over each balanced-batch along with common background-knowledge to obtain a large number of batch-wise plausible adverb/antonym indicator-rules over predicate-properties (such as those rules shown in Figure 2a). After all such single-time-step batch-plausible indicator-rules have been learnt for each adverb vs antonym task across the training data, we use those symbolic ASP rules to summarize object behaviours. Specifically, for an object's collection of behaviour-facts, we assign a 1 for an indicator-rule if that rule logically-fires for the given object's behaviour-facts, and we assign a 0 otherwise - so that from our collection of indicator-rules we obtain a vector of 0s and 1s (such as [1,1,1,0,1,1,...]) for each object-behaviour. All object-behaviours are converted in this manner for each adverb vs antonym task, and those vectors are used as rough behaviour-summaries for downstream adverb-type recognition. (Implementation details are further discussed in Appendix Section 6.3).

3.2.2 Transformer-Based Reasonoing

As an alternative to our single-time-step based symbolic-reasoning, we also propose multi-time-step transformer-based reasoning. We start by flattening the ASP-format object-behaviour properties detected by our Extraction Phase (as shown in Figure 3). We get rid of unnecessary syntactical detail and special characters that might otherwise confuse a sentence tokenizer, and record object-type only once per time-step to avoid redundancy. We also eliminate the explicit time-stamps (1, 2, 3...) associated with each logical fact. We are able to do this, provided that we maintain the correct chronological ordering of detected object-properties since transformer models already have provisions allowing them to recognize and reason over the positional-ordering of words in sentences.

```
detected(person, 0). magnitude(person, 0, efive_to_ninty).
angle(person, 0, e). sqsize(person, 0, verysmall).
celloccupancy(person, 0, 0, right, top).
celloccupancy(person, 0, 1, left, bottom).
celloccupancy(person, 0, 2, right, bottom).
operationarea(person, 0, large). movementinplace(person, 0, huge).
detected(person, 1). magnitude(person, 1, efive_to_ninty)...
```

(a) A truncated object-behaviour snippet showing ASP facts of object-behaviour for a 'person' type object across two time steps - formatted for symbolic-based reasoning.

person magnitude efivetoninty angle e sqsize verysmall celloccupancy zero right top celloccupancy one left bottom celloccupancy two right bottom operationarea large movementinplace huge person magnitude ...

(b) A truncated flattened representation of the object behaviour from Figure 3a - formatted for transformer-based reasoning.

Figure 3: Object Behaviour for a 'person' type object formatted for symbolic (a) and transformer (b) based reasoning.

Next, we consider Masked Language Modeling (MLM) [4] over object-behaviours to learn useful object-behaviour representations (Figure 4). In conventional MLM, some of the words of a naturallanguage sentence are masked out and a transformer model fitted with a shallow prediction-layer is trained to predict those masked words from the rest of the unmasked sentence - forcing the transformer to learn to encode sentence-structure and overall sentence meaning. Features output by the last transformer-layer are then typically extracted and used for related down-stream tasks such as text-classification. In the context of our object-behaviours problem setting, we directly extend this idea, by masking out some of the 'value-words' that correspond to each object's particular behaviour (that might be a value of magnitude/angle/operation-area/etc. at some time-steps). We then train an MLM transformer model to predict those masked values from the rest of the unmasked objectbehaviour (as shown in Figure $\frac{4}{2}$). In doing so, we force the transformer to learn to encode some overall meaning or dynamics of object-behaviour. The features output by the last transformer-layer are then used for down-stream adverb-type recognition. In particular, we do not train the transformer model from scratch, but rather fine-tune a model that has been pretrained for natural-language MLM. We do this transfer-learning in order to exploit complex network reasoning properties that have already been learnt over very large datasets of natural language³. Once we have fine-tuned our MLM model to reason over and unmask object-behaviours, we then use it to extract object-behaviour summary vectors. For each input object-behavior snippet in the dataset, we feed that flattened object-behavior to our trained transformer-model and extract the word-level vectors output by the transformer's final layer. Those word-level features are then averaged across the entire flattened sentence to give us a single summary vector - that encodes some overall multi-time-step object-behaviour information.

²Specifically, we mask value-words with a probability of 20% and do not mask-out prompt-words such as 'magnitude' and 'angle' that occur in every example. Importantly, we also make sure <u>not</u> to mask object-types as they can be difficult to infer from object-motion, and forcing a model to predict them would detract from learning other behavioural-properties.

³Note: to limit the computational costs of fine-tuning, we truncate flat object behaviour inputs (Figure 3b) at 512 tokens.



Figure 4: Masked Language Modeling over object-behaviours.

3.3 Prediction Phase

Finally, each summary object-behaviour feature vector (output by either the single-time step symbolicreasoning approach or the multi-time step transformer-reasoning approach) is then fed into a separate Support-Vector Machine (SVM) with rbf kernel for binary classification between each adverb-type and it's antonym. At test time, the adverb-vs-antonym predictions from multiple object-behaviours detected to be of interest in a single clip are aggregated to make a single decision by a majority-vote.

4 Experiments

Datasets: We evaluate our method on subsets of the MSR-VTT [23] and ActivityNet [2] datasets, using adverb-annotations by Doughty et al. [6]. We process clips where both raw-footage and adverbannotations are available using our Extraction Phase (Section 3.1), to obtain 1309 ASP-programs for our new MSR-VTT-ASP dataset and 1751 ASP-programs for our new ActivityNet-ASP dataset where each program contains facts of multiple object-behaviours detected to be of interest within the corresponding video-clip, along with background knowledge of predicate properties (Appendix 6.1). Each program is labeled with one or more of 22 adverb-types (11 adverb/antonym pairs) according the source clip's labels⁴: (1)upwards/downwards, (2)forwards/backwards, (3) outdoor/indoor, (4) slowly/quickly, (5)gently/firmly, (6)out/in, (7) partially/completely, (8)properly/improperly, (9) periodically/continuously, (10) instantly/gradually, (11) off/on. This leaves us with 1674 unique (asp-program, adverb) pairs from MSR-VTT and 1824 unique (asp-program, adverb) pairs from ActivityNet. We randomly split these datasets into training and testing sets using 70/30 stratified splits (stratified by adverb-type) to obtain 1171 training and 503 testing samples for MSR-VTT-ASP and 1276 training and 548 testing samples for ActivityNet-ASP. Finally, with these two new ASP-datasets and splits having been created, for experiments, we turn to the requirements of our adverb-type recognition framework. We require snippets of individual object-behaviours to reason-over for adverbtype prediction. So, for each ASP-program, we cut-out behaviour snippets for separate detected object-types - so that one snippet corresponds to one object-type's behaviour over the course of a video (as shown in Figure 3a). Each object behaviour snippet is annotated with the adverb-type of it's source program.⁵ These snippets of object-behaviour are then repeated within each adverb-category to balance out the number of samples used for training and testing in each category.

⁴We drop the loudly/quietly category since neither our method nor the previous work uses a clip's audio.

⁵Note: As each video-clip contains multiple objects, the number of object-behaviour snippets is much larger than the number of video-clips.

4.1 Symbolic Based Reasoning

As mentioned in section Section 3.2.1, we learn a large number of batch-wise plausible indicator rules using FastLAS over balanced batches of object-behaviours from the training-set within each adverb-vs-antonym category, and use those learnt indicator-rules to extract summary vectors of object behaviors for each adverb-vs-antonym classification task. Those summary vectors from the training-set are then used to train separate SVM classifiers to distinguish between each adverb and its antonym. At test time, SVM predictions from multiple object-behaviours within individual source-video-clips from the test-set are aggregated by majority-vote to distinguish between adverbs and antonyms in each category. As shown in Figure 5, the accuracy of prediction of this single-time-step based symbolic method is highest for both MSR-VTT and ActivityNet datasets when distinguishing between forwards-and-backwards type adverbs - which might plausibly be inferred from a grouping of single-time-step behaviour-properties. Performance is worst (zero) for more complex adverb-types: periodically-continuously, instantly-gradually and off-on - for which no single-time-step batch-wise-plausible range rules are found. Table 1 shows averaged accuracies across all adverb/antonym categories in each dataset.

4.2 Transformer Based Reasoning

For our experiments, we consider two light-weight versions of the landmark BERT [4] transformer model - namely the ALBERT [12] and DistilBERT [17] architectures. As outlined in Section 3.2.2, we fine-tune each pre-trained transformer model by flattening object-behaviour snippets and making them unmask randomly masked 'value-words'. We then obtain behaviour summary-vectors for each object-snippet by feeding their flat representations to the trained transformers without masking and averaging word-level features output by the last hidden layer across the entire sentence. As with the symbolic-case, a separate SVM with rbf kernel is used over these extracted summary-vectors, along with majority-voting to distinguish between each adverb and it's antonym. We find that both ALBERT and DistilBERT achieve comparable average-performance, while out-performing our symbolic approach by a wide margin (Table 1). However, when each adverb-vs-antonym recognition task is viewed separately (Figure 5), results are mixed. The symbolic approach performs best in the 'forward/backward' category, while one or the other of our transformer-based methods works best for other adverbs. The general superiority of the transformer approach is largely to be expected, given that it jointly reasons over multiple time-steps and multiple predicate properties, while our symbolic approach composes single time-step, single predicate properties. It can be difficult to interpret why one reasoning method outperforms another within a given category. However, it is encouraging that not all reasoning models exhibit the same performance, since we can achieve higher overall accuracy by separately using the most appropriate reasoning method for each category - as shown in Table 1.

4.3 Comparison with State-of-the-Art

We next make a strict comparison between our approaches and the action-dependent previous state-of-the-art [5, 6]. For previous methods, we randomly flip action-type labels for 5% of action-categories in the train and test sets, so that we obtain 'imperfect-actions' that represent an action-type prediction-accuracy of 95% (which one might approach if a state-of-the-art action-type predictor [22] is trained on full-versions of the two datasets and used for prediction). In the case of MSR-VTT-ASP, our joint Symbolic and Transformer action-free reasoning method is highly competitive, making a 3.71% improvement over PsudoAdverbs[6] in the imperfect-actions case and outperforming previous works even in the scenario where all ground truth actions are explicitly known (Table 1). In the case of ActivityNet-ASP, our joint Symbolic and Transformer action-free reasoning method achieves 1.86% lower accuracy than PsudoAdverbs in the imperfect-actions-case, but is still highly useful, as it offers an action-free alternative to previous methods at a relatively small drop in performance - with no requirement to train or maintain action-type predictors.

Broader Impact: Similar to action-type recognition methods, we reflect that one may attempt to use adverb-type recognition to maliciously interpret and monitor video-footage. However, we also note that improved adverb-type recognition, when used ethically for improved video-interpretation, offers significant benefits to human computer interaction and robotics.

Limitations and Scope for Future Work: To our knowledge, we are the first to propose



Figure 5: Binary classification accuracies for adverb-vs-antonym tasks on the ASP test-sets of balanced object-behaviours using our proposed symbolic and transformer-based approaches.

Method	MSR-VTT-ASP Average accuracy (%)↑	ActivityNet-ASP Average accuracy (%) ↑
ActionModifiers[5] w/ ground-truth-actions PsudoAdverbs[6] w/ ground-truth-actions	59.98 67.49	61.05 67.59
ActionModifiers[5] w/ imperfect-actions PsudoAdverbs[6] w/ imperfect-actions	59.78 64.61	59.88 65.76
Ours (Action-Free):		
Symbolic-Based Prediction	57.08	36.51
ALBERT Transformer-Based Prediction	64.10	61.26
DistilBERT Transformer-Based Prediction	64.48	60.65
DistilBERT + ALBERT Prediction	67.92	62.96
Symbolic + DistilBERT + ALBERT Prediction	<u>68.32</u>	<u>63.90</u>

Table 1: Average adverb-vs-antonym binary-classification accuracies using our proposed <u>action-free</u> reasoning methods, and the <u>action-dependent</u> previous state-of-the-art on the balanced ASP test-sets and corresponding clips (ground-truth actions-available scenario greyed out).

action-free methods and first to reason-over object behaviours for adverb-type recognition. As such there are several possible directions for further investigation. Primarily, for transformers we explored MLM-modeling using light-weight transformer models, and our symbolic-reasoning method is limited to single-time step and single-predicate type rules. Scope for future work then includes exploring alternative transformer-modeling/architectures (such as causal modeling using GPT-3 [1]), and reasoning over multiple-time steps and multiple-predicate-properties using symbolic-reasoning.

5 Conclusion

In this work, we proposed the design of a new framework that reasons over object-behaviours to recognize a video-clip's adverb-types. Importantly, unlike previous work, our method is action-free and is directly applicable when the action-type of a video-clip is unknown. We proposed a novel pipeline to extract human-interpretable object-behaviour-facts from raw video clips and used that pipeline to create two new datasets of object-behaviour-facts - the MSR-VTT-ASP and ActivityNet-ASP datasets. Finally, we proposed novel symbolic and transformer based reasoning methods that reason over those extracted facts to distinguish between adverb/antonym types. Experiment results demonstrate that our proposed methods perform favourably against the previous state-of-the-art.

References

- [1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020.
- [2] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 961–970, 2015.
- [3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6299–6308, 2017.
- [4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [5] Hazel Doughty, Ivan Laptev, Walterio Mayol-Cuevas, and Dima Damen. Action modifiers: Learning from adverbs in instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 868–878, 2020.
- [6] Hazel Doughty and Cees GM Snoek. How do you do it? fine-grained action understanding with pseudoadverbs. arXiv preprint arXiv:2203.12344, 2022.
- [7] Haodong Duan, Yue Zhao, Yuanjun Xiong, Wentao Liu, and Dahua Lin. Omni-sourced webly-supervised learning for video recognition. In *European Conference on Computer Vision*, pages 670–688. Springer, 2020.
- [8] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003.
- [9] Shreyank N Gowda, Marcus Rohrbach, and Laura Sevilla-Lara. Smart frame selection for action recognition. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 1451–1459, 2021.
- [10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In Proceedings of the IEEE international conference on computer vision, pages 2961–2969, 2017.
- [11] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pages 1725–1732, 2014.
- [12] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942, 2019.
- [13] Mark Law, Alessandra Russo, Elisa Bertino, Krysia Broda, and Jorge Lobo. Fastlas: Scalable inductive logic programming incorporating domain-specific optimisation criteria. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2877–2885, 2020.
- [14] Vladimir Lifschitz. Answer set programming. Springer Berlin, 2019.
- [15] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Álayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 2630–2640, 2019.
- [16] Bo Pang, Kaiwen Zha, and Cewu Lu. Human action adverb recognition: Adha dataset and a three-stream hybrid model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2325–2334, 2018.
- [17] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108, 2019.
- [18] Zheng Shou, Xudong Lin, Yannis Kalantidis, Laura Sevilla-Lara, Marcus Rohrbach, Shih-Fu Chang, and Zhicheng Yan. Dmc-net: Generating discriminative motion cues for fast compressed video action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1268–1277, 2019.
- [19] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. *Advances in neural information processing systems*, 27, 2014.
- [20] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.
- [21] Xin Wang, Jiawei Wu, Junkun Chen, Lei Li, Yuan-Fang Wang, and William Yang Wang. Vatex: A large-scale, high-quality multilingual dataset for video-and-language research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4581–4591, 2019.
- [22] Wenhao Wu, Zhun Sun, and Wanli Ouyang. Revisiting classifier: Transferring vision-language models for video recognition. *Proceedings of the AAAI, Washington, DC, USA*, pages 7–8, 2023.
- [23] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5288–5296, 2016.

6 Appendix

6.1 Implementation Details of the Extraction Phase

In this section, we describe the design of our extraction-phase in greater detail. Figure 6 shows a depiction of this pipeline.

ASP- representation of object facts recorded for each window as a separate time-step.



Delayed Capture Frames

Figure 6: A depiction of our video-to-discrete object behaviours pipeline.

As mentioned earlier, given a raw video clip, we first employ MaskRCNN [10] over delayed-captures of static frames from the video clip's image sequence - the delay is added so that we only consider every fifth frame of the original clip and avoid processing immediately successive frames (between which very little changes). MaskRCNN gives us a collection of predicted object-types and bounding boxes along with their corresponding confidence scores between 0 and 1. We ignore all detections made with a confidence score less than 0.3, and flag all detected patches with low confidence scores between 0.3 and 0.5 as 'unknown' object-types. Patches detected with a confidence score above 0.5 are recorded along with their predicted object-types.

To capture properties of motion for each of these detected object patches, we compute the pixel-wise Gunnar-Farneback optical flow [8] between consecutive delayed capture frames. These per-pixel optical-flow values are averaged within each detected object bounding box to give us a single numeric value of optical-flow magnitude and a single numeric value of optical-flow angle for each detected object-patch.

To filter these numerous detections for the most adverb-relevant information, we make two important assumptions.

- First, we assume that of the objects detected in a scene, faster moving objects are of more interest for adverb-type recognition then slower moving objects within the same scene. This is a reasonable assumption to make since we are trying to design a system that mimics human judgement of adverb recognition in video clips, and to humans, faster moving objects are usually more eye-catching and take precedence over slower-moving objects.
- Next, we make the assumption that objects whose behaviour determine the video clip's overall adverb-type must be detected to be of interest (moving faster than other objects within the same scene) with some level of consistency. If an object is deemed to be of interest only fleetingly, then it is unlikely to determine the overall categorization of a video clip.

Acting on these two assumptions, after computing averaged optical-flow properties for each bounding box as described above, we then run a non-overlapping sliding window over the delayed capture frames and filter out objects that (1) Do not have optical-flow magnitude above the average of all objects detected in the same frame (2) Do not pass the first filtering step for at least half the delayed-capture frames encompassed by the sliding window. In our implementation, we use a sliding window of size five -within which period, the types of objects being portrayed do not usually change much.

Finally, to simplify the tracking of object behaviour between frames, we ignore duplicate object-types detected within each delayed frame and record only the object with the highest optical-flow magnitude in a contest between two or more objects of the same type. Once we have filtered our MaskRCNN detections this way, the per-frame properties of optical-flow magnitude, optical-flow angle and bounding-box size for objects are averaged across all detections of the same object-type within each window. Since scenes do not usually change much within the span of a window, these object-properties that we are averaging for a given object-type usually pertain to the same physical object.

Properties recorded from each window correspond to a separate time-step and are timestamped as ASP facts accordingly. As shown in Figure 6, "detected(person, 3)" means that an object of type 'person' is detected at time-step 3 - corresponding to the third window scan.

In addition to these averaged per-frame properties, we also capture local temporal properties of 'operation-area' and 'movement-in-place' for each object within a window. Operation-area captures the size of the area within which a single object-type 'lives' for the span of a window i.e it is the product of (xmax-xmin) and (ymax-ymin) computed over all detected bounding box coordinates. Movement-in-place on the other hand is the ratio of the operation-area to average bounding-box-size of the detected object within a window. The more that an object moves around within a given window, the larger that this ratio will be. To simplify the down-stream reasoning process, all numeric properties besides optical-flow magnitude are placed into discrete buckets, such as 'small', 'very-small', 'medium', etc. ; while angles are categorized into discrete sectors 'north(n)', 'north-east(ne)', 'east(e)' and so on. The exact numerical-range of each discrete bucket/sector is specified in the accompanying code-implementation.

Besides the object propeties already mentioned, we also record the region of the video frame within which each object's operation-area is maximum (i.e the section of the frame within which the object 'lives'). Rather than mapping each frame's area using a series of flat grid cell locations such as $C_0, C_1, C_2, C_3...$ and assigning cell-ids to each object, in order to allow a more intuitive reasoning process, we employ a hierarchy of relative placement.

As shown in Figure 7, at the highest level of this hierarchy (level 0), the frame is split into 4 regions, (top, left), (top, right), (bottom, left) and (bottom, right). At the second level (level 1), each of these regions is further split into another 4 regions, and again at level 2, the process is repeated. Describing the location of an object's operation-area using this hierarchy of relative placement then allows us to make fairly simple inferences. For example, if level 1 placement stays



Figure 7: Hierarchy of relative-placement for cell-occupancy information.

unchanged and the level 2 toggles from left to right, we can easily determine that the object has moved right by a small amount, and if instead the level 1 placement toggles from top to bottom, then we can say that the object has moved downward by a significant amount. As shown in Figure 6, cell-occupancy predicates map the operation-area of detected objects in each window using this hierarchy of relative-placement.

Finally, for clarity, Figure 8 shows a trimmed example of an ASP-program, computed from a video-clip using this pipeline. The figure shows some selected background knowledge and object-properties detected over a single time step. Importantly, we highlight that our background knowledge includes information on the 'opposites' of relative directions, as well as 'less-than', 'clockwise' and 'anticlockwise' orderings over bucketed numeric values - so that we can reason about ranges in symbolic approaches. Importantly, these ordering predicates are formulated with a measure of distance to allow for range-reasoning. For example, 'clockwise(n, ne, 1)' indicates that northeast is clockwise of north by one-tick while 'clockwise(n, e, 2)' indicates that east is clockwise from north by two-ticks. Similarly 'less-than(very-small, small, 1)' indicates that small is larger than very-small by one step, whereas 'less-than(very-small, medium, 2)' indicates that medium is larger than very-small by two steps.

6.2 MSR-VTT-ASP and ActivityNet-ASP datasets

As discussed in Section 4, in this work, using our extraction-phase pipeline, we process clips where both raw-footage and adverb -annotations (from Doughty et al. [6]) are available for the MSR-VTT[23] and ActivityNet[2] video-datasets to create the new MSR-VTT-ASP and ActivityNet-ASP datasets of ASP-programs. Each dataset contains facts of multiple-object behaviours detected to be of interest within the corresponding video clip, along with background knowledge of predicate properties - as shown in Figure 8 (the full background-knowledge of predicate-properties are specified in the ASP-files of the new datasets). As discussed in Section 4, each program is labeled with one or more of 22 adverb-types (11 adverb/antonym pairs) according the source clip's labels: (1)upwards/downwards, (2)forwards/backwards, (3) outdoor/indoor, (4) slowly/quickly, (5)gently/firmly, (6)out/in, (7) partially/completely, (8)properly/improperly, (9) periodically/continuously, (10) instantly/gradually, (11) off/on, and we split these datasets into 70/30 train/test stratified splits (stratified by adverb-type) to obtain 1171 training and 503 testing samples for MSR-VTT-ASP and 1276 training and 548 testing samples for ActivityNet-ASP. Table 2 shows summary properties of these two new datasets.

			Average.				
	# Train	# Test	# obj per. clip	# time-steps per. clip	# obj per. time-step		
MSR-VTT-ASP	1171	503	3.06	16.35	0.93		
ActivityNet-ASP	1276	548	3.24	29.61	0.87		

Table 2: Summary of properties of the MSR-VTT-ASP and ActivityNet-ASP datasets - Ignoring 'unknown' types and assuming that an object-type corresponds to the same object across all detections.

```
% Selected Background Knowledge on Directions, Sizes and Orderings---
opposites(left, right).
opposites(top, bottom).
opposites(right, left).
opposites(bottom, top).
clockwise(n, ne, 1).
clockwise(ne, e, 1).
anticlockwise(nw, n, 1).
anticlockwise(w, nw, 1).
less_than(very_small, small, 1).
less_than(small, medium, 1).
less_than(A, C, P+Q) :- less_than(A, B, P), less_than(B, C, Q).
clockwise(A, C, P+Q) :- clockwise(A, B, P), clockwise(B, C, Q), P+Q<8.
anticlockwise(A, C, P+Q) :- anticlockwise(A, B, P), anticlockwise(B, C, Q), P+Q<8.
% Objects in clip------
object(sports_ball).
object_property(sports_ball, inanimate_movable).
object(person).
object_property(person, person).
object(unknown).
object_property(unknown, unknown).
% Properties detected in time-step 0 -----
detected(person, 0).
magnitude(person, 0, 44).
angle(person, 0, e).
sq_size(person, 0, large).
cell_occupancy(person, 0, 0, left, top).
cell_occupancy(person, 0, 1, right, bottom).
cell_occupancy(person, 0, 2, left, bottom).
operation_area(person, 0, large).
movement_in_place(person, 0, very_small).
```

Figure 8: Trimmed example of object-behaviour captured in an ASP program.

6.3 Implementation Details of Symbolic-Based Reasoning

As mentioned in Section 3.2.1, we consider employing the FastLAS inductive learning method [13] to automatedly learn some governing rules over these extracted facts - so as to explain the overall adverb-type categorization of each video clip.

In using FastLAS to learn such rules, we are primarily constrained by the number and type of variables that each rule can use. The more number of variables that a rule allows, the more the number of possible groundings that it can take, and the longer that it takes for rule learning to complete. And while each extracted instance of object behaviour possess multiple facts of the same predicates across different time-steps (such as magnitude at time step 1, magnitude at time step 2, etc.), owing to the large number of possible groundings, automatedly learning rules that reason over

more than one time-step is especially challenging for this task.

In this work, rather than attempting to overcome these challenges and learn a few very complex rules over multiple time-steps and multiple predicate-properties, we instead consider learning a large number of simpler rules, that compositionally might inform overall adverb-type.

To limit the number of free variables that FastLAS has to deal with, we focus on learning range-rules that define upper or lower bounds of an object's predicate-properties at single time-steps. To illustrate this idea, Figure 9 shows a toy example that we feed into FastLAS. The example specifies the behavior four objects: a car, a plane, a person, and a cat. Each of these behaviors has a corresponding optical flow magnitude for an arbitrary time-step and each object-behaviour (that is a positive example for our rule-learning problem) is also associated with a particular class type - either 'strange' or 'not-strange'. The first class-type mentioned in a #pos header in the figure is the one that we wish to associate with the object-behaviour. The second class type mentioned in the header is what the object-behaviour is <u>not</u>.

As shown in this toy example, in order to reduce the number of variable-values that Fast-LAS deals with, we also use discretized versions of optical-flow magnitude such as 'five-to-ten', 'ten-to-fifteen', etc.

The language-bias shown in the figure specifies that the head atom of any learnt rule must be a class type ('strange' or 'not-strange'), and must generalize over variable objects-types. The language-bias also specifies that body atoms (if used) must capture some range property over magnitude. As FastLAS can learn to use one or none of each of the specified 'less-than' body atoms, a learnt-rule might enforce an upper bound on magnitude value, a lower bound on magnitude value, or neither. Additionally, as the 'number-of-steps' field in the less-than predicate is specified as a FastLAS numeric-variable (num-var), FastLAS is allowed to learn numeric-constraints that further explain range-rules.

For this particular toy example, we can explain all of the provided object-behaviours by deeming magnitudes between 5 and 20 to correspond to the class 'strange' and other magnitude values to correspond to the class 'not-strange'. FastLAS does infact discover such corresponding rules, as shown in Figure 10. Figure 11 shows a depiction of these learnt ranges for better clarity.

We can similarly employ these range-style language-biases for other predicate properties such as optical-flow angle, and operation area. Figure 12 shows how we might do so. As also shown in Figure 12, for cell-occupancy we consider using a slightly different language bias - we allow for rule-body conditions that consist of: (A)A variable relative direction along the vertical (top/bottom) and a constant horizontal direction (left/right), (B)A variable direction along the horizontal (left/right) and a constant vertical direction (top/bottom) or (C)Both horizontal and vertical relative directions specified as constants. We also use a numeric-variable (num-var) value for the level of cell-hierarchy used by a rule, so that we can learn rules that apply to different levels.

An important facet of this type of observational-predicate rule learning is that it specifically requires numeric rule learning (either to learn constraints over the number-of-steps range property or the level-of-hierarchy as described), and we have chosen to employ FastLAS as it is the only framework that allows this type of automated numeric-rule learning over ASP programs.

Generalizing our toy example for the more complex problem-setting of recognizing adverb-types from object-behaviours is straightforward. We use recorded object behaviours from our video-to-ASP pipeline as positive examples in the rule-learning setup, wherein the class associated with each object-behaviour is the ground-truth adverb-type of the overall video clip that an object hails from. Naturally, the class <u>not</u> to be associated with each object behaviour is the antonym of its adverb-type. Figure <u>13</u> shows a truncated example of a detected object's behaviour formatted for FastLAS rule-learning. To simplify our explorations and to reduce noise in the training data, we ignore 'unknown' object-behaviours detected by our pipeline.

However, problems arise in using this learning methodology as-is over sets of object-behaviours for given (adverb, antonym) pairs. Firstly if the set of object-behaviours is not-balanced to have an equal

```
% Trimmed Background Knowledge -----
object(person). object(cat). object(car). object(plane).
typ(not_strange). typ(strange).
time(1..25). count(1..10).
speed(zero_to_five). speed(five_to_ten). # and so on.
less_than(zero_to_five, five_to_ten, 1).
less_than(five_to_ten, ten_to_fifteen, 1). # and so on.
less_than(A, C, P+Q) :- less_than(A, B, P), less_than(B, C, Q).
% Examples for FastLAS ------
#pos(eg(id0)@100, { class(strange, car) }, { class(not_strange, car) }, {
  magnitude(car, 20, five_to_ten).
}).
#pos(eg(id1)@100, { class(strange, plane) }, { class(not_strange, plane) }, {
  magnitude(plane, 20, ten_to_fifteen).
}).
#pos(eg(id2)@100, { class(not_strange, person) }, { class(strange, person) }, {
  magnitude(person, 19, zero_to_five).
}).
#pos(eg(id3)@100, { class(not_strange, cat) }, { class(strange, cat) }, {
  magnitude(cat, 23, twenty_to_twfive).
}).
% START LANGUAGE BIAS ------
#maxv(3).
#modeh(class(const(typ), var(object))).
#modeb(1, magnitude(var(object), var(time), var(speed))).
#modeb(1, object(var)).
#modeb(1, time(var)).
#modeb(1, speed(var)).
#modeb(1, less_than(var(speed), const(speed), num_var(count))).
#modeb(1, less_than(const(speed), var(speed), num_var(count))).
% Scoring Function ------
#bias("penalty(1, head(X)) :- in_head(X).").
#bias("penalty(1, body(X)) :- in_body(X).")
```

Figure 9: Toy example of range-based rule-learning.

Figure 10: Rules learnt by FastLAS for the toy-example shown earlier.



Figure 11: A depiction of the ranges defined by the rules learnt for our toy-example.

number of objects for both adverb and antonym, then we might get best coverage by just predicting a single class rule without any body conditions. This problem of unbalanced data is easily solved by repeating object-behavior examples in the training data to balance out the adverb/antonym classes. The next problem is more important - since the adverb-recognition setting is quite noisy (with many of our detected object-behaviours not necessarily impacting a video's overall adverb-type), for large sets of object-behaviour, even after balancing, we find that rules learnt from our simple language biases (Figure 12) are not able to cover more than 50% of behaviours. Then, directly predicting one or another adverb/antonym head with no body conditions again becomes the best strategy for maximum coverage.

As mentioned earlier, increasing the complexity of possible rules to abate this problem comes with its own set of challenges (namely rule-learning can slow down to the point of becoming computationally impractical). So, rather than increasing the complexity of our language biases, we consider smaller-batches of balanced subsets of the training data - which pose a less noisy and less complex problem setting when each batch is viewed separately. We run FastLAS separately over each balanced-batch of object behaviours to get batch-wise plausible rules for each of our language-biases. When the rules returned by FastLAS for a batch and language bias are non-trivial (possess body-conditions), we then record them as indicators of adverb-type.

After all such indicator-rules have been extracted from a stream of balanced-batches of the full training set for each (adverb, antonym) pair using our language-biases (Figure 12), we then consider composing their results together using Support Vector Machines (SVMs). Specifically, given an input object-behaviour and an adverb vs antonym task, we assign a 1 to each corresponding indicator-rule if the rule fires for the given object's behaviour, and assign a 0 otherwise - so that from our collection of indicator-rules of the adverb/antonym task, we obtain for the object-behaviour a feature of 1s and 0s (such as [1,1,1,0,1,1,...]). The entire balanced training-set is converted in this manner for each adverb vs antonym task. A separate binary-SVM with rbf kernel is trained over these extracted features to classify between every adverb and its antonym.

At inference time, a raw video clip is converted to an ASP program of object behaviors, and all the indicator-rules are checked to obtain vectors of zeros and ones for each object. All the SVMs make their adverb/antonym predictions (over features from their corresponding indicator-rules)

```
% LANGUAGE BIAS FOR OPTICAL-FLOW MAGNITUDE------
\#maxv(3).
#modeh(class(const(typ), var(object))).
#modeb(1, magnitude(var(object), var(time), var(speed))).
#modeb(1, object(var)).
#modeb(1, time(var)).
#modeb(1, speed(var)).
#modeb(1, less_than(var(speed), const(speed), num_var(count))).
#modeb(1, less_than(const(speed), var(speed), num_var(count))).
% LANGUAGE BIAS FOR OPTICAL-FLOW ANGLE------
#modeh(class(const(typ), var(object))).
#modeb(1, object(var)).
#modeb(1, time(var)).
#modeb(1, direction(var)).
#modeb(1, angle(var(object), var(time), var(direction))).
#modeb(1, clockwise(var(direction), const(direction), num_var(count))).
#modeb(1, clockwise(const(direction), var(direction), num_var(count))).
% LANGUAGE BIAS FOR OPERATION-AREA -----
#maxv(3).
#modeh(class(const(typ), var(object))).
#modeb(1, operationarea(var(object), var(time), var(size))).
#modeb(1, object(var)).
#modeb(1, time(var)).
#modeb(1, size(var)).
#modeb(1, less_than(var(size), const(size), num_var(count))).
#modeb(1, less_than(const(size), var(size), num_var(count))).
#maxv(4).
#modeh(class(const(typ), var(object))).
#modeb(1, celloccupancy(var(object), var(time), num_var(count),
                      const(relative_direction), var(relative_direction))).
#modeb(1, celloccupancy(var(object), var(time), num_var(count),
                      var(relative_direction), const(relative_direction))).
#modeb(1, celloccupancy(var(object), var(time), num_var(count),
                      const(relative_direction), const(relative_direction))).
#modeb(1, object(var)).
#modeb(1, time(var)).
#modeb(2, relative_direction(var)).
```

Figure 12: Language Biases used for rule-learning over optical-flow magnitude, optical-flow angle, operation-area and cell-occupancy.

```
#pos(eg(id0)@100, { class(adverb, person) }, { class(antonym, person) }, {
    detected(person, 0). magnitude(person, 0, efive_to_ninty).
    angle(person, 0, e). sqsize(person, 0, verysmall).
    celloccupancy(person, 0, 0, right, top).
    celloccupancy(person, 0, 1, left, bottom).
    celloccupancy(person, 0, 2, right, bottom).
    operationarea(person, 0, large). movementinplace(person, 0, huge).
    detected(person, 1). magnitude(person, 1, efive_to_ninty)
    angle(person, 1, e). sqsize(person, 1, verysmall).
    celloccupancy(person, 1, 0, left, top).
    celloccupancy(person, 1, 2, left, bottom).
    operationarea(person, 1, large). movementinplace(person, 1, huge). ....}).
```

Figure 13: A truncated example of a detected object-behaviour formatted for FastLAS rule-learning.

for each detected object, and predictions from multiple objects detected within a single-clip are aggregated by a simple voting mechanism in each adverb vs antonym category.

6.4 Compute Requirements for Experiments

All experiments presented in this work can be reproduced using a single P5000 GPU device. Using this resource, ASP-Program facts were extracted for the full MSR-VTT-ASP and ActivityNet-ASP datasets sequentially over 2 days, while transformer-based finetuning completes in under 1 hr for a given dataset for both DistilBERT and ALBERT architecures. Learning rules from balanced-batches of object-behaviours using FastLAS and a single CPU requires roughly 20 hours to complete for each dataset. All SVM training and inference completes in under 5 min.