Contents

1 Introduction

Writing algorithms which can answer questions on pictures or videos with a high level of accuracy and generality has been a goal of researchers in the AI community for many years. Recently, a lot of progress has been made in this area; with advances in neural network models and the production of larger datasets allowing reseachers to significantly improve accuracy on question answering models.

Formally, Visual Question Answering (VQA) [3] is a task where, given an image and a question posed in natural language about the image, a model is required to produce an open-ended answer to the question. Video Question Answering (VideoQA) is a related task where a model is given a video (mulitple images in sequence) and a question. These questions can be related to a single frame of the video, effectiively making VideoQA a superset of the VQA task.

This project will attempt to produce a hybrid model for VideoQA - one which makes use of both neural networks and knowledge representation and reasoning methods based on first-order logic.

1.1 Motivation

VQA and VideoQA tasks attract attention because of their difficulty; both problems are considered "AI-complete" - they require knowledge from multiple modalities beyond a single sub-domain [2]. Building systems which have a deep understanding of the world would be a significant achievement for AI research; allowing many tasks which require significant human time and effort to be automated. Solving the VideoQA problem, which require image understanding, natural language understanding and commensense reasoning to be deployed, could be a major step towards this.

Furthermore, subproblems of VideoQA have already been shown to have applications in real-world tasks, for example event recognition has been used for identifier attacks on computer networks [8], detecting credit card fraud [32] and recognising cardiac arrhythmias [27].

Finally, the use of a hydrid model for VideoQA brings with it a nmuber of advantages. Firstly, representing knowledge in logical form allows the injection of commensense or background knowledge which can significantly improve accuracy in question answering tasks [25]. Secondly, there has recently been a significant increase in research related to explainable AI methods - machine learning techniques that enable human users to understand, trust and manage emerging artificially intelligent partners [4]. Extracting the knowledge from a neural network into logical form could be an important step toward explaining and understanding their behaviour.

1.2 Problem Description

As mentioned above, the VideoQA problem can be defined as building a model which, when presented with a short video and an open-ended, natural language question about the video, can produce a natural language answer to the given question. In our case we are looking to design a hybrid model for VideoQA. More specifically, convolutional neural networks (CNNs) will be used to extract knowledge from each frame of the video. This knowledge, along with the question to be answered, will be represented in a fashion that is amenable to searching for the answer to the question. This framing of the problem leads us to outline the following sub-problems, which will need to solved in order to produce a satisfactory VideoQA model.

1. **Object Detection**. Given a frame, we need a model which can produce a rough estimate (a bounding box, for example) of the location of an object in the frame. We will also need a model which can classify each detected object into a set of predefined classes.

2. **Property Extraction**. Given an object, which is the output of the 'object detection' model above, we need an algorithm which can produce a set of values for that object for some set of predefined properties. For example, we might need to give a value for the colour, size or shape of an object.

3. **Event Detection**. Given two sequential frames, we need a model which can classify the event(s) which occurred between the two frames into a set of predefined classes (possibly including a catch-all 'no event' class). This model will also be required to list objects involved in the event and what their role in the event was. This will require some level of object tracking so that it is clear how objects are related between frames.

4. **Question and Knowledge Representation**. Given the outputs of the models above and the natural language question, we need a way of representing the background knowledge, the question and the knowledge contained in the frames of the video. These must be represented in a manner that allows an answer to the question to be found.

1.3 Objectives

A lot of research has been conducted on building end-to-end neural network models to solve VideoQA tasks (see section 3 for examples), but very little prior work has been done on hybrid models. The main aim of this project, therefore, is to explore the possibility of adding logical representation of knowledge to existing deep learning techniques. The following are the primary objectives of the project:

1. Construct a hybrid VideoQA model which allows injection of background knowledge and helps to increase the explainability of the model.
2. Find a challenging dataset for training the model (or construct a dataset if none are suitable)
3. Compare qualitatively (and quantitatively, if possible) existing approaches to our own hybrid model.
4. Investigate the possibility of learning domain-dependent logical rules, rather than having to have them hard coded for each environment.

2 Background

This section introduces some technical background which will likely be required as part of the project. It includes an introduction to neural networks and CNNs, a comparison of some existing object tracking and object detection algorithms and a discussion on knowledge representation and reasoning and symbolic rule learning.

2.1 Deep Learning

Deep neural networks (DNNs) have emerged as a very successful algorithm for machine learning; deep learning has been used to beat records in tasks such as image recognition, speech recognition and language translation [22]. Many different architectures have been proposed to solve various tasks, these architectures include convolutional neural networks (CNNs), which are designed to process data that come in the form of multiple arrays [22], and recurrent neural networks (RNNs), which are designed to process sequences of arbitrary length [24]. The following section gives a brief introduction to CNNs and describes some of their use cases.

2.1.1 Convolutional Neural Networks

CNNs contain three types of layers: convolution, pooling and fully connected. Units (artificial neurons) in a convolution layer are organised into feature maps. The inputs to each unit in a feature map come from the outputs of the units in a small region of the previous layer, the output of the unit is then calculated by passing the weighted sum of its inputs through an activation function such as ReLU. The set of weights, also known as a filter or kernel, is the part of the layer which is learned through backpropagation. Every unit in a feature map has the same kernel. Each feature map in a layer has its own kernel. Pooling layers reduce the size of the input by merging multiple units into one. A typical pooling operation is max-pooling, which computes the maximum of a local patch of units. Finally, in fully-connected

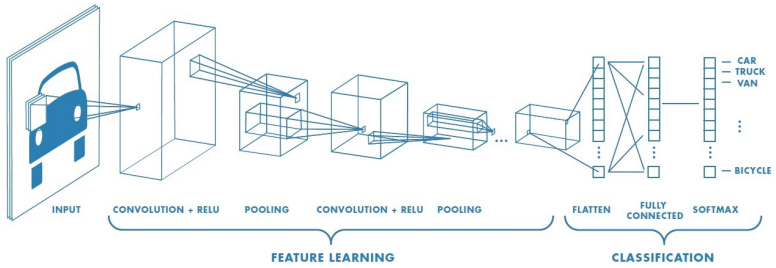


Figure 1: An example of a CNN architecture. The input image is passed through a series of convolution and pooling layers before being flattened into a one-dimensional layer and passed through one final fully connected layer. The softmax classification function is then applied at the output.

layers (which are typically placed at the output of the CNN) every unit in a layer is connected to every unit in the previous layer. An example CNN architecture is shown in figure 1.

CNNs have proven to be adept at a number of tasks involving images, including image classification [15] and object detection [28][30]. We explore these further in Section 2.2.

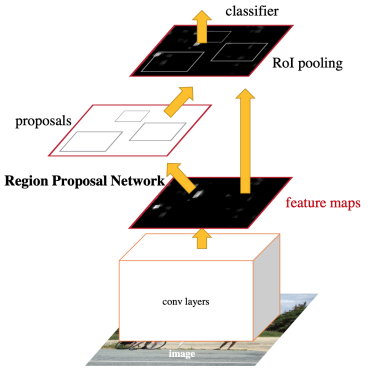
2.2 Image Processing

2.2.1 Object Detection

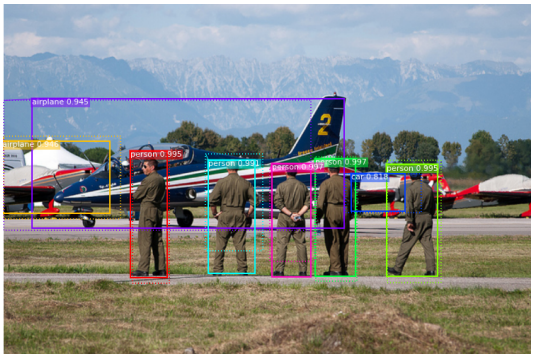
The object detection task could formally be defined as designing a model which, when given an image, can produce a rough localisation of objects of interest in the image (in the form of a bounding box) and classify eac of these objects into a set of predefined classes. In this section we introduce two well known object detection algorithms, Faster R-CNN [30] and You Only Look Once (YOLO) [28].

Faster R-CNN is an evolution of previous object detection algorithms, R-CNN [12] and Fast R-CNN [11]. Faster R-CNN builds on its predecessors by adding a region proposal network (RPN) - a neural network which takes an image and produces a set of region of interest (RoI) proposals. This method of region proposal is much faster than previous algorithms (such as those used in [12] and [11]) since it is able to make use of the GPU, as opposed to requiring the CPU. Faster R-CNN then uses a similar classifier and bounding box regressor as Fast R-CNN at the output; this section of the network also receives the feature maps from the final layer of the RPN, in this sense the initial layers of the network are shared between the region proposal section and the classifier/regressor section. A diagram of the Faster R-CNN architecture is shown in figure 2a.

The three object detection algorithms mentioned above all work by first producing region proposals, then producing a more accurate localisation and



(a) Diagram of the Faster R-CNN architecture. Figure from [30].



(b) An example of the bounding boxes and confidence scores produced by an object detection algorithm. Image from [1].

Figure 2

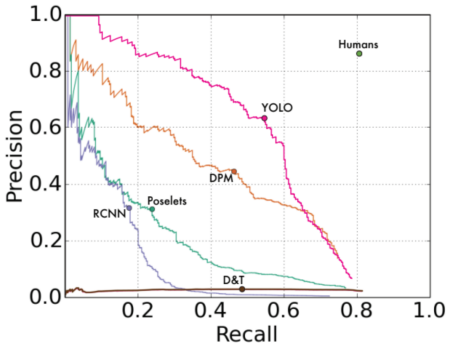
a score for each region and finally removing any low-scoring or redundant regions. This requires the algorithm to 'look' at the image multiple times (around 2000 times for R-CNN). You Only Look Once (YOLO) is a significantly more time-efficient algorithm which, as the name suggests, takes a single look at the image. A convolutional neural network is used to simultaneously predict multiple bounding boxes and the class probabilities for each box. As well as being very fast, YOLO makes fewer than half the number of background errors (where the algorithm mistakes background patches for objects) as Fast R-CNN [29]. YOLO is, however, slightly less accurate than some of the slower methods for object detection [28].

2.2.2 Commonly Used Metrics

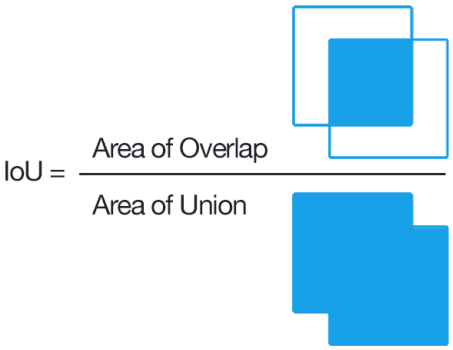
In this section we present some commonly used metrics for classification and object detection tasks. We use TP, TN, FP and FN to mean True Positive, True Negative, False Positive and False Negative, respectively.

Firstly, for classification tasks the following terminology is commonly used:

- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$. The accuracy is the ratio of correct predictions to the total number of predictions.
- $Precision = \frac{TP}{TP+FP}$. The precision is the ability of a classifier to not label the negative data as positive.
- $Recall = \frac{TP}{TP+FN}$. The recall is the ability of the classifier to find the positively-labelled data.
- $F_1 = 2 * \frac{precision*recall}{precision+recall}$. The F_1 score is a way of combining the precision and recall scores.



(a) Example precision-recall curves for various object detection models. Image from [29].



(b) Visual explanation of the intersection over union metric. Image from [31].

Figure 3: Precision-recall curves and definition of intersection over union

Each object detector model will output a confidence score for each object classification it makes. We can then set a threshold value which determines what is counted as a classification of an object. Altering this threshold value will give different precision and recall values for the model, which can then be plotted on a precision-recall graph. Example precision-recall curves are shown in figure 3a.

For object detection tasks, where a bounding box is produced as a rough localisation of an object's position, metrics which measure the accuracy of the localisation of an object are required. One very common metric is the Average Precision (AP), which is roughly defined as the area under the precision-recall curve (estimates of this value are usually used for competition datasets). In order to assess how well a model localises an object in an image the Interesection over Union (IoU) metric is commonly used. Figure 3b gives a visualisation on how IoU is calculated. Each IoU threshold will produce its own precision-recall curve.

2.2.3 Multiple Object Tracking

Multiple Object Tracking (MOT) is a computer vision task that aims to identify and track objects from a sequence of images without any prior knowledge about the appearance or number of targets [6]. Most object tracking methods share a very similar pipeline [6], as follows:

- **Detection**. Each input frame is analysed to identify objects using bounding boxes.
- **Feature Extraction**. Algorithms extract appearance, motion and/or interaction features from objects.

- **Affinity Computation**. Features are used to compute a similarity score between objects.
- **Association**. Similarity scores are used to associate detections and compute the object trajectories.

The Simple Online and Realtime Tracking (SORT) [5] algorithm is one of the best performers [6]. It makes use of Faster R-CNN for object detection, the Kalman filter [14] framework for predicting object motion, IoU as a similarity function and the Hungarian algorithm [16] for association. SORT attempts to track objects using motion prediction, other algorithms, however, focus on extracting features from objects, such as Histogram of Orientated Gradients (HOG) [7] features, and using these to match objects in successor frames. More recent methods tend to use features extracted directly from a CNN.

2.3 Knowledge Representation and Reasoning

Knowledge Representation and Reasoning (KRR) is concerned with how intelligent agents store and manipulate their knowledge. In this section we discuss Answer Set Programming, a logic programming framework, action languages, which can be used to define the behaviour of a system, and methods for learning logical rules inductively.

2.3.1 Answer Set Programming

Answer Set Programming (ASP) is a form of declarative logic programming that can be used to solve difficult search problems. Whereas imperative programs define an algorithm for finding a solution to a problem, logic programs simply define a problem, it is then the job of logic program solvers to find a solution to the problem. ASP also differs from Prolog, another popular logic programming language, in that ASP programs are purely declarative. This means that reordering rules or atoms within rules has no effect on the output of the solver [9]. ASP solvers work by finding the answer sets of the program, where each rule in the program imposes restrictions on what can be an answer set. An answer set can be thought of as a set of ground atoms which satisfies every rule of the program (although the full definition of an answer set is too in-depth for this discussion).

The following templates are a some of the possible forms of rules in an ASP program:

$$a \text{ :- } b_1, ..., b_k, not\ b_{k+1}, ..., not\ b_m. \tag{1}$$

$$\{c_1; ...; c_n\} \text{ :- } b_1, ..., b_k, not\ b_{k+1}, ..., not\ b_m. \tag{2}$$

$$\text{ :- } b_1, ..., b_k, not\ b_{k+1}, ..., not\ b_m. \tag{3}$$

Where each a, b_i and c_i are atoms in first-order logic. The left hand side of the rule is known as the head, and the right hand side is known as the body. The *not* in the rule body stands for negation-as-failure, which means that *not* b_i will be satisfied when b_i cannot be proved. Each rule requires that when the body is satisfied - that is, when every member of the body is satisfied - the head must be satisfied. Rule 2 is known as a choice rule. The head of a choice rule can be satisfied by any subset of the atoms inside the brackets (including the empty set); in effect, a choice rule creates possible answer sets. Finally, rule 3 is known as a constraint. The body of a constraint must not be satisfied; intuitively, a constraint rules out answer sets.

As well as negation as failure, ASP also has a notion of 'strong negation'. The strong negation of an atom p is written $\neg p$. Strong negation can be thought of as logical negation, although it does not always have the same properties. In practice, ASP solvers implement strong negation by treating $\neg p$ as an additional atom, and enforce that no answer set can contain both p and $\neg p$.

2.3.2 The Action Language $\mathcal{AL}$

Action languages are formal models for describing the behaviour of dynamic systems. In this section we present the version of $\mathcal{AL}$ given in [10]. $\mathcal{AL}$'s signature contains three special sorts: *statics*, *fluents* and *actions*. Fluents are partitioned into two sorts: *inertial* and *defined*. Statics and fluents are both referred to as 'domain properties'. A 'domain literal' is a domain property or its negation. Statements in $\mathcal{AL}$ can be of the following form:

$$a \textbf{ causes } l_{in} \textbf{ if } p_0, ..., p_m \tag{4}$$

$$l \textbf{ if } p_0, ..., p_m \tag{5}$$

$$\textbf{impossible } a_0, ..., a_k \textbf{ if } p_0, ..., p_m \tag{6}$$

Where a is an action, l and $p_0, ..., p_m$ are domain literals and l_{in} is a literal formed by an inertial fluent. Statement 4 is known as a *causal law*, 5 as a *state constraint* and 6 as an *executability condition*. A collection of $\mathcal{AL}$ statements is known as a 'system decription'. An $\mathcal{AL}$ system description can be used to model the behaviour of dynamic systems with discrete states; each state can be seen as the set of fluents which are true and transitions between states are caused by actions.

It also possible to encode a given $\mathcal{AL}$ system description, along with a number of 'domain-independent' axioms, in ASP. The method for creating this encoding is given in Appendix A.

2.3.3 Symbolic Rule Learning

Inductive Logic Programming [26] (ILP) is a field of symbolic AI research concerned with learning symbolic rules which, when combined with background knowledge, entail a set of positive examples and do not entail any negative examples. ILASP [21] (Inductive Learning of Answer Set Programs) is an ILP framework for learning ASP programs.

The authors of [17] define the *Learning from Answer Sets* (ILP_{LAS}) task (which is the task solved by the original version of ILASP), by first defining a *partial interpretation*. A partial interpretation E is a pair of sets of atoms E^{inc} and E^{exc}, known as the *inclusions* and *exclusions* of E. An answer set A *extends* E if it contains all of the inclusions ($E^{inc} \subseteq A$) and none of the exlusions ($E^{exc} \cap A = \emptyset$). An ILP_{LAS} task is then defined as the tuple $T = \langle B, S_M, E^+, E^- \rangle$, where B is the background knowledge, S_M is the search space, E^+ and E^- are the partial interpretations for the positive and negative examples, respectively. An hypothesis H is known as an inductive solution of T if and only if all of the following are true:

1. $H \subseteq S_M$
2. $\forall e^+ \in E^+ \; \exists A \in AS(B \cup H)$ *such that* A *extends* e^+
3. $\forall e^- \in E^- \; \nexists A \in AS(B \cup H)$ *such that* A *extends* e^-

Where $AS(P)$ refers to the answer sets of a program P.

Later versions of ILASP are capable of solving more complex tasks, including learning weak constraints [20] (a method for specifying preferences in ASP), learning from context dependent examples [19] and learning from noisy examples [18].

3 Related Work

3.1 VideoQA Datasets

A number of datasets are available for the VideoQA problem, in this section we discuss each of the available datasets.

The MovieQA dataset [33] is a VideoQA dataset consisting of 14,944 multiple-choice questions which align with video clips from movies. The clips come from a collection of 408 movies and the Question-Answer (QA) pairs were generated by humans in three steps. Firstly, one annotator would be given a plot summary for a moive and asked to create correct QA pair, additionally this annotator would mark the area of the plot summary to which their QA pair corresponded. Next, a different annotator would be asked to provide four incorrect answers to a QA pair (and was given the option to correct the original question and answer). Finally, annotators

would align each sentence in the plot summary to the video by marking the start and end points, this allowed each QA pair to be aligned with a video clip. The questions and anwers are written in free-form natural language.

Zeng et al. [34] create a much larger VideoQA dataset by automatically generating QA pairs from videos and associated descriptions collected online. Their dataset consists of 18100 videos as well 151263 and 21352 automatically generated QA pairs in the training and validation sets, respectively. The dataset also contains 2461 human-generated QA pairs to be used for testing. Their questions and answers are free-form natural language, however, a large number of their answers are yes and no (32.5% and 32.5%, respectively).

The TGIF-QA dataset [13] is commonly used for assessing the performance of VideoQA models. The updated version of the TGIF-QA dataset contains human-generated 165,165 QA pairs collected from 71,741 GIFs, sourced from the TGIF dataset [23], which contains a number of GIFs and associated descriptions. There are four possible types of questions in the TGIF-QA dataset, three specific to VideoQA, requiring temporal knowledge to answer, and the last asks questions related to a single frame (VQA). The question types are as follows:

- **Repetition Count**. Counting the number of repetitions of an action. There are 11 possible answers (0 to 10+).
- **Repeating Action**. A multiple-choice question about identifying an action that has been repeated in the video. For example, **Q**: What does the duck do three times? **A**: Shake head.
- **State Transition**. A multiple-choice question about identifying the state before or after another state. For example, **Q**: What does the bear do after sitting? **A**: Stand.
- **FrameQA**. Open-ended questions related to a single frame.

For the VideoQA questions the authors created templates for questions and used a large number of human annotators to speed up the generation process. The FrameQA questions are generated using the descriptions from the TGIF dataset. A number of quality control checks were also included.

4 Project Plan

References

[1] W. Abdulla. *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. URL: `https://github.com/matterport/Mask_RCNN` (visited on 01/15/2020).

[2] Somak Aditya, Yezhou Yang, and Chitta Baral. "Explicit Reasoning over End-to-End Neural Architectures for Visual Question Answering". In: *CoRR* abs/1803.08896 (2018). arXiv: `1803.08896`. URL: `http://arxiv.org/abs/1803.08896`.

[3] Aishwarya Agrawal et al. "Vqa: Visual question answering". In: *International Journal of Computer Vision* 123.1 (2017), pp. 4–31.

[4] Alejandro Barredo Arrieta et al. *Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI*. 2019. arXiv: `1910.10045 [cs.AI]`.

[5] Alex Bewley et al. "Simple online and realtime tracking". In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2016, pp. 3464–3468.

[6] Gioele Ciaparrone et al. "Deep learning in video multi-object tracking: A survey". In: *Neurocomputing* (2019).

[7] Navneet Dalal and Bill Triggs. "Histograms of Oriented Gradients for Human Detection". In: *International Conference on Computer Vision & Pattern Recognition (CVPR '05)*. Vol. 1. IEEE Computer Society, 2005, pp. 886–893.

[8] Christophe Dousson, Pierre Le Maigat, and France Telecom R&d. "Chronicle Recognition Improvement Using Temporal Focusing and Hierarchization". In: *IJCAI*. 2007, pp. 324–329.

[9] Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner. "Answer Set Programming: A Primer". In: *Reasoning Web. Semantic Technologies for Information Systems: 5th International Summer School 2009*. Ed. by Sergio Tessaris et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 40–110.

[10] Michael Gelfond and Yulia Kahl. *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*. Cambridge University Press, 2014.

[11] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[12] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[13] Yunseok Jang et al. “Tgif-qa: Toward spatio-temporal reasoning in visual question answering”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2758–2766.

[14] Rudolph Emil Kalman et al. “A new approach to linear filtering and prediction problems [J]”. In: *Journal of basic Engineering* 82.1 (1960), pp. 35–45.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105.

[16] Harold W Kuhn. “The Hungarian method for the assignment problem”. In: *Naval research logistics quarterly* 2.1-2 (1955), pp. 83–97.

[17] Mark Law, Alessandra Russo, and Krysia Broda. “Inductive Learning of Answer Set Programs”. In: *Logics in Artificial Intelligence*. Springer International Publishing, 2014, pp. 311–325.

[18] Mark Law, Alessandra Russo, and Krysia Broda. “Inductive learning of answer set programs from noisy examples”. In: *arXiv preprint arXiv:1808.08441* (2018).

[19] Mark Law, Alessandra Russo, and Krysia Broda. “Iterative learning of answer set programs from context dependent examples”. In: *Theory and Practice of Logic Programming* 16.5-6 (2016), pp. 834–848.

[20] Mark Law, Alessandra Russo, and Krysia Broda. “Learning weak constraints in answer set programming”. In: *Theory and Practice of Logic Programming* 15.4-5 (2015), pp. 511–525.

[21] Mark Law, Alessandra Russo, and Krysia Broda. *The ILASP system for learning Answer Set Programs*. `www.ilasp.com`. 2015.

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521 (2015), pp. 436–444.

[23] Yuncheng Li et al. “TGIF: A new dataset and benchmark on animated GIF description”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4641–4650.

[24] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. “Recurrent neural network for text classification with multi-task learning”. In: *arXiv preprint arXiv:1605.05101* (2016).

[25] Kenneth Marino et al. “OK-VQA: A Visual Question Answering Benchmark Requiring External Knowledge”. In: *CoRR* abs/1906.00067 (2019). arXiv: `1906.00067`.

[26] Stephen Muggleton. “Inductive logic programming”. In: *New generation computing* 8.4 (1991), pp. 295–318.

[27] Ren Quiniou et al. “Intelligent Adaptive Monitoring for Cardiac Surveillance”. In: *Computational Intelligence in Healthcare 4: Advanced Methodologies*. Ed. by Isabelle Bichindaritz et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 329–346.

[28] Joseph Redmon and Ali Farhadi. “Yolov3: An incremental improvement”. In: *arXiv preprint arXiv:1804.02767* (2018).

[29] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[30] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[31] Adrian Rosebrock. *Intersection over Union (IoU) for object detection*. Nov. 7, 2016. URL: `https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection` (visited on 01/15/2020).

[32] Nicholas Poul Schultz-Møller, Matteo Migliavacca, and Peter Pietzuch. “Distributed Complex Event Processing with Query Rewriting”. In: *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*. DEBS 09. New York, NY, USA: Association for Computing Machinery, 2009.

[33] Makarand Tapaswi et al. “MovieQA: Understanding Stories in Movies through Question-Answering”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 4631–4640.

[34] Kuo-Hao Zeng et al. “Leveraging Video Descriptions to Learn Video Question Answering”. In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI Press, 2017, 43344340.

Appendix A ASP encoding of $\mathcal{AL}$

In this appendix we present a method for encoding an $\mathcal{AL}$ system description in ASP, as described in [10]. We need to encode three different parts of the system description: the signature, the $\mathcal{AL}$ statements and some domain-independent axioms.
We encode the signature of the system description, $sig(SD)$, as follows:

- For each constant symbol c of sort $sort\_name$ other than $fluent$, $static$ or $action$, $sig(SD)$ contains

$$sort\_name(c). \tag{7}$$

- For every static g of SD, $sig(SD)$ contains

$$static(g). \tag{8}$$

- For every intertial fluent f of SD, $sig(SD)$ contains

$$fluent(inertial, f). \tag{9}$$

- For every defined fluent f of SD, $sig(SD)$ contains

$$fluent(defined, f). \tag{10}$$

- For every action a of SD, $sig(SD)$ contains

$$action(a). \tag{11}$$

In the following we refer to the ASP encoding of the $\mathcal{AL}$ system description as $\Pi(SD)$, where $\Pi(SD)$ includes $sig(SD)$. We introduce a relation $holds(f, i)$ which says that fluent f is true at timepoint i. We also introduce the notation $h(l, i)$ where l is a domain literal and i is a step, which will not be used in the ASP program, but will instead be replaced by either $holds(f, i)$ if $l = f$, or by $\neg holds(f, i)$ if $l = \neg f$.
We encode the $\mathcal{AL}$ statements as follows:

- If the maximum number of steps is $< max >$, then $\Pi(SD)$ includes

$$\#const\ n = \ < max > . \tag{12}$$
$$step(0..n). \tag{13}$$

- For every causal law, a **causes** l **if** $p_0, ..., p_m$ $\Pi(SD)$ contains

$$h(l, I + 1) \text{ :- } h(p_0, I), ..., h(p_m, I), occurs(a, I), I < n. \tag{14}$$

- For every state constraint, l **if** $p_0, ..., p_m$ $\Pi(SD)$ contains

$$h(l, I) \text{ :- } h(p_0, I), ..., h(p_m, I). \tag{15}$$

- For every executability condition, **impossible** $a_0, ..., a_k$ **if** $p_0, ..., p_m$ $\Pi(SD)$ contains

$$\neg occurs(a_0, I); ...; \neg occurs(a_k, I) \text{ :- } h(p_0, I), ..., h(p_m, I). \tag{16}$$

The ; in the head of rule 16 stands for logical disjunction and can be read as meaning at least one of $a_0, ..., a_k$ must not occur at timepoint I.

In order to complete our encoding of an $\mathcal{AL}$ system description we need to add a number of domain-independent axioms. These axioms are not specific to any system or task, but rather convey commonsense knowledge that should apply to many systems. It is worth noting, however, that in certain situations some or all of these axioms may not make sense and should not be used.
We encode the domain-independent knowledge as follows:

- The inertia axiom states that inertial fluents will keep their state unless explicitly changed:

$$\begin{aligned} holds(F, I+1) \text{ :- } & fluent(inertial, F), \\ & holds(F, I), \\ & not\ \neg holds(F, I+1), \\ & I < n. \end{aligned} \tag{17}$$

$$\begin{aligned} \neg holds(F, I+1) \text{ :- } & fluent(inertial, F), \\ & \neg holds(F, I), \\ & not\ holds(F, I+1), \\ & I < n. \end{aligned} \tag{18}$$

- The closed-world assumption (CWA) for defined fluents states that defined fluents which are not known to be true are assumed to be false.

$$\neg holds(F, I) \text{ :- } fluent(defined, F), step(I), not\ holds(F, I). \tag{19}$$

- The CWA for actions states that actions that are not known to occur are assumed to not occur.

$$\neg occurs(A, I) \text{ :- } action(A), step(I), not\ occurs(A, I). \tag{20}$$

Finally, in order to make use of system description encoding we would include information on which events occured, using the $occurs(A, I)$ predicate, and information on the different states of the system using $holds(F, I)$ and $\neg holds(F, I)$.