Department of Computing

Imperial College of Science, Technology and Medicine

Learning and Reasoning with Spacio-Temporal Properties for Understanding Scene Sequences

Interim Report

Author:
Ross Irwin

Supervisors:
Prof. Alessandra Russo
Dr. Krysia Broda

January 22, 2020

Contents

Chapter 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) [4] 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, effectively making VideoQA a superset of the VQA task.

This project will attempt to produce a hybrid model for VideoQA - one which is capable of learning spatial relations through neural networks and symbolic rules through inductive logic programming.

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. In particular, we will investigate learning spatial relations within frames using neural networks and learning a symbolic model of an environment using inductive logic programming.

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 [3]. 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 requires 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 [12], detecting credit card fraud [47] and recognising cardiac arrhythmias [42].
Finally, the use of a hybrid model for VideoQA brings with it a number 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 [39]. 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 [5]. 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 Chapter 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.

Chapter 2

Background

This chapter 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 [32]. 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 [32], and recurrent neural networks (RNNs), which are designed to process sequences of arbitrary length [35]. 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 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 2.1.

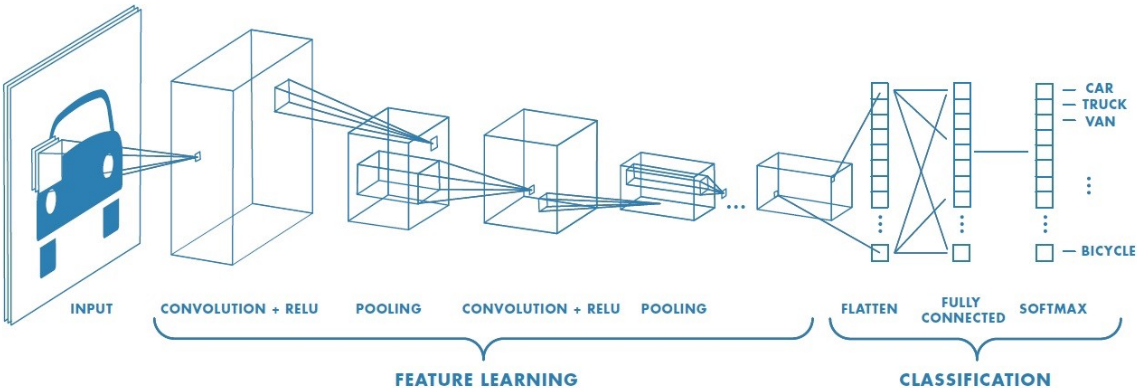


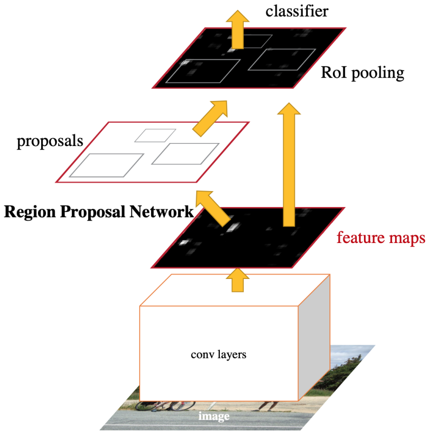
Figure 2.1: An example of a CNN architecture. The input image is passed through a series of convolution and pooling layers before being flattened into a one-dimensional layer and passed through one final fully connected layer. The softmax classification function is then applied at the output.

CNNs have proven to be adept at a number of tasks involving images, including image classification [25] and object detection [43, 45]. We explore these further in Section 2.2.

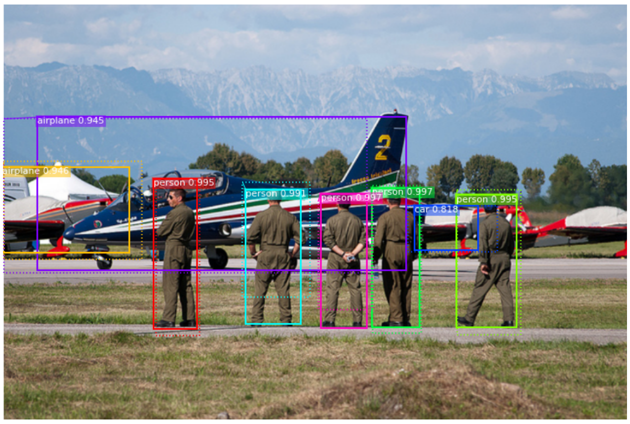
2.2 Image Processing

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 each of these objects into a set of predefined classes. In this section we introduce two well known object detection algorithms, Faster R-CNN [45] and You Only Look Once (YOLO) [43].
Faster R-CNN is an evolution of previous object detection algorithms, R-CNN [19] and Fast R-CNN [18]. 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 [19] and [18]) 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 2.2a.
The three object detection algorithms mentioned above all work by first producing region proposals, then producing a more accurate localisation and 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,



(a) Diagram of the Faster R-CNN architecture. Figure from [45].



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

Figure 2.2

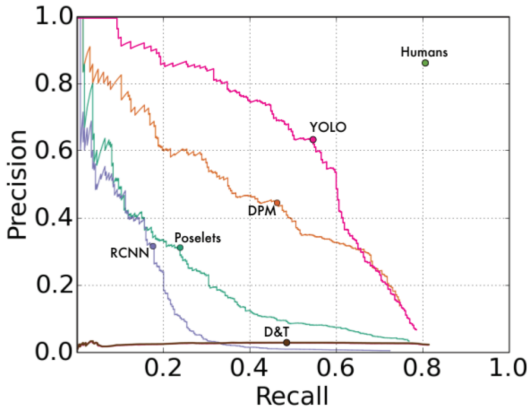
as the name suggests, takes a single look at the image. A convolutional neural network is used to simultaneously predict multiple bounding boxes and the class probabilities for each box. As well as being very fast, YOLO makes fewer than half the number of background errors (where the algorithm mistakes background patches for objects) as Fast R-CNN [44]. YOLO is, however, slightly less accurate than some of the slower methods for object detection [43].

2.2.2 Commonly Used Metrics

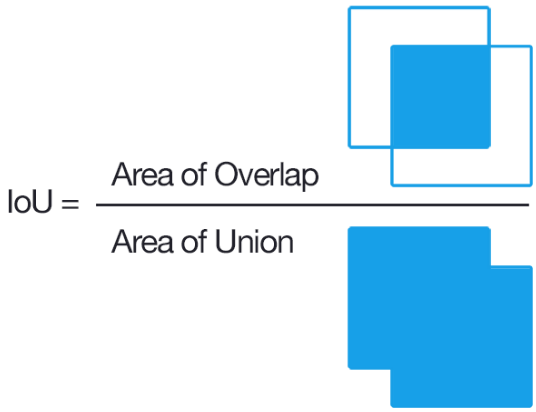
In this section we present some commonly used metrics for classification and object detection tasks. We use TP, TN, FP and FN to mean True Positive, True Negative, False Positive and False Negative, respectively.
Firstly, for classification tasks the following terminology is commonly used:

- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$. 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.

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 2.3a.



(a) Example precision-recall curves for various object detection models. Image from [44].



(b) Visual explanation of the intersection over union metric. Image from [46].

Figure 2.3: Precision-recall curves and definition of intersection over union

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 2.3b 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 [10]. Most object tracking methods share a very similar pipeline [10], 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) [8] algorithm is one of the best performers [10]. It makes use of Faster R-CNN for object detection, the Kalman filter [23] framework for predicting object motion, IoU as a similarity function and the Hungarian algorithm [26] for association. Whereas SORT attempts to track objects using motion prediction, other algorithms focus on extracting features from objects and using these to match objects in successor frames. Commonly used features include Histogram of Orientated Gradients (HOG) [11], Scale Invariant Feature Transform (SIFT) [36] descriptors and Speeded-Up Robust Features (SURF) [7] descriptors. 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 [13]. 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 some of the possible forms of rules in an ASP program:

$$a \text{ :- } b_1, ..., b_k, not\ b_{k+1}, ..., not\ b_m. \tag{2.1}$$

$$l\{c_1; ...; c_n\}u \text{ :- } b_1, ..., b_k, not\ b_{k+1}, ..., not\ b_m. \tag{2.2}$$

$$\text{:- } b_1, ..., b_k, not\ b_{k+1}, ..., not\ b_m. \tag{2.3}$$

Where a and each b_i and c_i are atoms in first-order logic, l and u are integers

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 2.2 is known as a choice rule. The head of a choice rule can be satisfied by any subset, S, of the atoms inside the brackets, provided $l \leq |S| \leq u$; in effect, a choice rule creates possible answer sets. Finally,

rule 2.3 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 classical 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 [17]. $\mathcal{AL}$'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:

$$a \textbf{ causes } l_{in} \textbf{ if } p_0, ..., p_m \tag{2.4}$$

$$l \textbf{ if } p_0, ..., p_m \tag{2.5}$$

$$\textbf{impossible } a_0, ..., a_k \textbf{ if } p_0, ..., p_m \tag{2.6}$$

Where:
- a is an action
- l and $p_0, ..., p_m$ are domain literals
- l_{in} is a literal formed by an inertial fluent

Statement 2.4 is known as a *causal law*, 2.5 as a *state constraint* and 2.6 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 is 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 [41] (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 [31] (Inductive Learning of Answer Set Programs) is an ILP framework for learning ASP programs.
The authors of [27] 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 [30] (a method for specifying preferences in ASP), learning from context dependent examples [29] and learning from noisy examples [28].

Chapter 3

Related Work

3.1 Datasets for VideoQA

A number of datasets are available for the VideoQA problem, in this section we discuss each of the available datasets. A comparison of all the datasets discussed in this section is shown in Table 3.1.

The MovieQA dataset [52] is a VideoQA dataset consisting of 14,944 multiple-choice questions about parts of movies. The clips come from a collection of 408 movies and the Question-Answer (QA) pairs were generated by humans. The questions and each of the possible answers are written in natural language, and there are five possible answers for each question.

Zeng et al. [60] create a much larger VideoQA dataset by automatically generating QA pairs from videos and their 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 [22] is commonly used for assessing the performance of VideoQA models. The dataset contains 165,165 human-generated QA pairs collected from 71,741 GIFs, sourced from the TGIF dataset [34], which contains a number of GIFs and associated descriptions. There are four possible types of questions in the TGIF-QA dataset, three of which are specific to VideoQA; requiring temporal knowledge to answer. The question types are as follows:

- **Repetition Count**. Counting the number of repetitions of an action. There are 11 possible answers (0, ..., 9, 10+).
- **Repeating Action**. A multiple-choice question about identifying an action that has been repeated in the video.
- **State Transition**. A multiple-choice question about identifying the state before or after another state.
- **FrameQA**. Open-ended questions related to a single frame.

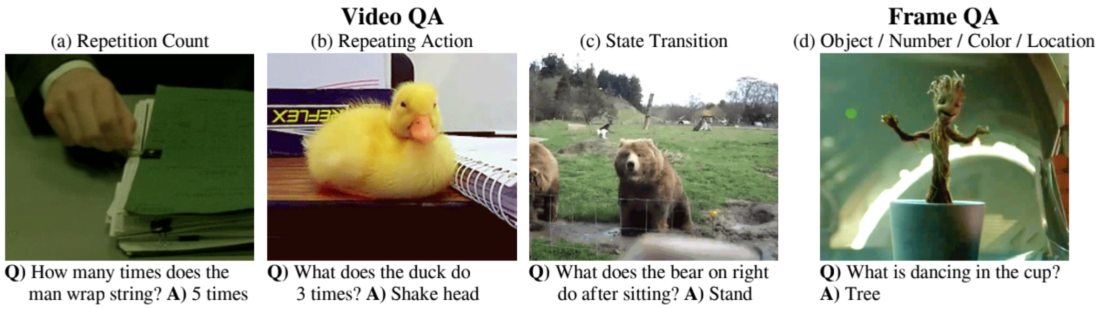


Figure 3.1: Example videos and QA pairs included in the TGIF-QA dataset, split by question type. Figure from [22].

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.

Zhu et al. [61] have proposed a VideoQA dataset containing fill-in-the-blank (FIB) style questions, with multiple-choice answers. The dataset contains over 100,000 real-world video clips and 400,000 questions. The dataset is generated from three different annotated video sources. On top of questions which ask the model to describe the present (describe the current video), for two of three video sources the authors also introduce two additional question types: infer the past and predict the future. For these two types of questions the model is asked a question on a part of the video which it is not explicitly given; these questions require the model to use some form of commonsense reasoning to generate a correct answer. One of the advantages of using a multiple-choice dataset, such as this, is that it is more amenable to quantitative evaluation than datasets with free-form answers, since answers are either right or wrong.

The EgoVQA dataset [14] attempts to address the lack of first-person VideoQA datasets. The dataset contains 581 QA pairs with both multiple-choice questions (with 5 possible answers per question) and open-ended questions. The dataset was created by manually generating QA pairs from a pre-existing set of 16 first-person videos. The authors also show that existing VideoQA models only marginally outperformed random choice on questions related to the colour of objects. They conjecture that existing models struggle to separate attentions on camera wearers from attentions on third persons.

Xu et al. [56] generate two VideoQA datasets by converting video captions into QA pairs. The first dataset, known as MSVD-QA, is generated from the Microsoft Research Video Description Corpus [9] which is used in many video captioning experiments. MSVD-QA contains 1,970 video clips and 50,505 QA pairs. Similarly, the second dataset, known as MSRVTT-QA, is generated from the MSR-VTT dataset [57]. The MSRVTT-QA dataset contains 10,000 video clips and 243,680 QA pairs.

The YouTube2Text-QA dataset [58] is another large dataset for VideoQA generated from a pre-existing video description dataset, in this case the YouTube2Text [20]

dataset is used. The YouTube2Text-QA dataset consists of 1,970 videos and 99,421 QA pairs.
The TVQA dataset [33] contains 21,793 video clips and 152,545 QA pairs based on 6 popular TV shows. The QA pairs were annotated manually using *Amazon Mechanical Turk*. Workers were asked to generate questions in the format: [What/How/Where/Why/Who/Other] _____ [when/before/after] _____. The second part of the question localises the relevant video moment within the clip, while the first part contains the question about that moment. The answers to the questions are given in multiple-choice format, with five candidate answers for each question.
The PororoQA dataset [24] is created from video clips and subtitles of the children's cartoon series '*Pororo*'. The dataset contains 8,913 multiple-choice QA pairs and 16,066 video clips.
Finally, the MovieFIB dataset [37] is a large-scale fill-in-the-blank style dataset generated from movie descriptions. The dataset contains 128,085 video clips and 348,998 QA pairs. The questions concern entities, actions and objects; answering these questions therefore implies that a model has some level of visual understanding of the scene, rather than being able to answer based purely on the given partial sentence. Answers are open-ended (not multiple choice) but each answer is only a single word.

Table 3.1: Comparison of discussed VideoQA datasets. Each row contains data on: the number of videos/clips, the number of QA pairs, whether the uses multiple-choice questions, whether the dataset uses fill-in-the-blank questions and the video source.

Dataset	#Videos	#QA pairs	MC	FIB	Source
MovieQA	408[1]	14,944	Y	N	Movies
Zeng et al.	18,100	175,076	N	N	Online videos
TGIF-QA	71,741	165,165	Y[2]	N	Online videos
Zhu et al.	>100,000	400,000	Y	Y	Various
EgoVQA	16	581	Y	N	First-person videos
MSVD-QA	1,970	50,505	N	N	Video desc. corpus
MSRVTT-QA	10,000	243,680	N	N	Video desc. corpus
Youtube2Text-QA	1,970	99,421	Y	N	YouTube videos
TVQA	21,793	152,545	Y	N	TV shows
PororoQA	16,066	8,913	Y	N	Cartoon series
MovieFIB	128,085	348,998	N	Y	Movie description

[1] Full length movies. Some of the QAs come with timestamps, allowing more video clips.

[2] FrameQA questions (53,083 QA pairs) are not MC.

3.2 VideoQA Implementations

All of the datasets described in the section above (except [14]) were presented along with original neural network models for solving the VideoQA task. This section attempts to summarise some of these approaches, along with a few others [16,

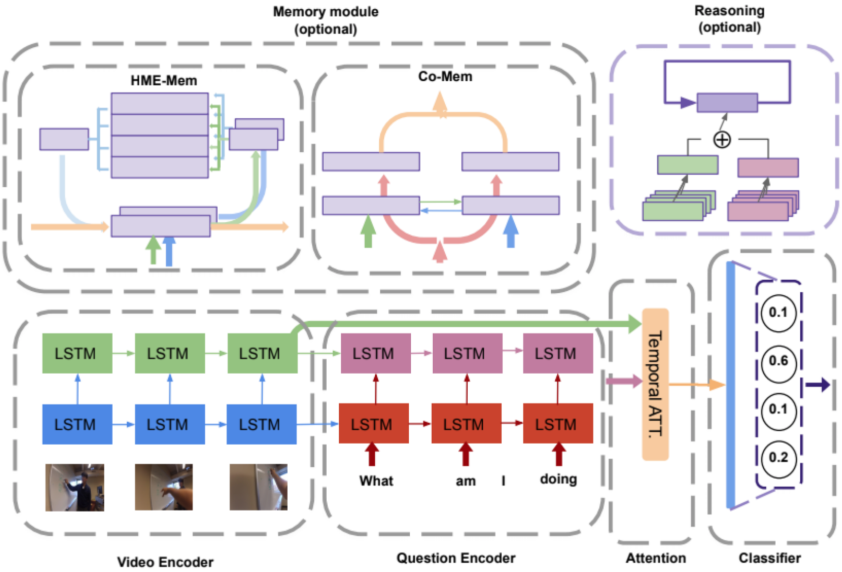


Figure 3.2: Typical VideoQA architecture. Figure from [14].

59, 15, 49]. However, since the focus of our approach will not be an end-to-end neural architecture, we do not give detailed descriptions. An example of a typical VideoQA neural architecture is given in Figure 3.2.

All previous work presented here contains a video encoder for extracting features from frames of the video. These usually include both appearance and motion features, which are extracted from pre-trained networks (ResNet [21], VGG [48] or GoogLeNet [51], for appearance, and C3D [53], for motion). The features extracted from each frame are then usually passed into LSTM or GRU networks to obtain encodings for the whole video.

Questions are often encoded by generating word embeddings for each word of the sentences and then passing the list of words to a sentence encoder, such as the LSTM or GRU architectures.

Visual attention [40] is used to help neural networks focus on the most relevent areas of an image or video. Applying attention mechanisms to associate a question with its most relevant frames (or areas within frames) has become a key part of more recent VideoQA models [22, 56, 58, 33, 16, 59, 15, 24]. Temporal attention is commonly applied to help the model focus on the most salient frames of the video, however, [22] applies both temporal and spatial attention, which also allows the model to attend to the most relevant regions of a frame.

3.3 External Knowledge for VQA

While VideoQA research has focused on end-to-end neural network architectures, some recent research in VQA (single frame setting) has experimented with using explicit reasoning layers and integrating external knowledge. A number of VQA datasets which require some form of external 'commonsense' reasoning have been proposed recently [38, 55, 54]. It has been shown that end-to-end neural networks which do not attempt to make use of knowledge which is external to the training data perform poorly on some of these datasets [38]. This section discusses some of the attempts that have been made to integrate external knowledge into VQA systems.

The authors of the three datasets outlined above propose models which make use of external knowledge, ususally stored in some structured knowledge base (KB), such as DBpedia [6], which stores structured information extracted from Wikipedia, or ConceptNet [50], which contains automatically generated 'common-sense' relations between objects.

As opposed to using a structured KB, the authors of [38] provide a neural network model which is trained to find the answer to an image-question pair from Wikipedia articles. They also propose a number of methods for combining this network with state-of-the-art VQA models and show that this provides an improvement in performance on their dataset.

The authors of [2] propose a model which first extracts properties from an image using a pre-trained neural network and represents these properties explicitly using logic. They also extract relations between nouns, adjectives and the question word from the question and represent these in logic. Finally, they reason over the extracted relations using a probabilistic reasoning engine to find the most likely answer. This method of reasoning not only allows the model to make use of external knowledge, but also helps improve the transparency and explainability of the model.

Chapter 4

Project Plan

Bibliography

[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: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[3] 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).

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

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

[6] Sören Auer et al. “Dbpedia: A nucleus for a web of open data”. In: *The semantic web*. Springer, 2007, pp. 722–735.

[7] Herbert Bay et al. “Speeded-up robust features (SURF)”. In: *Computer vision and image understanding* 110.3 (2008), pp. 346–359.

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

[9] David L Chen and William B Dolan. “Collecting highly parallel data for paraphrase evaluation”. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics. 2011, pp. 190–200.

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

[11] 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.

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

[13] 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. Springer Berlin Heidelberg, 2009, pp. 40–110.

[14] Chenyou Fan. "EgoVQA - An Egocentric Video Question Answering Benchmark Dataset". In: *The IEEE International Conference on Computer Vision (ICCV) Workshops*. 2019.

[15] Chenyou Fan et al. "Heterogeneous Memory Enhanced Multimodal Attention Model for Video Question Answering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1999–2007.

[16] Jiyang Gao et al. "Motion-appearance co-memory networks for video question answering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6576–6585.

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

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

[19] 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.

[20] Sergio Guadarrama et al. "Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition". In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 2712–2719.

[21] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[22] 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.

[23] 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.

[24] Kyung-Min Kim et al. "DeepStory: Video Story QA by Deep Embedded Memory Networks". In: IJCAI17. AAAI Press, 2017, 20162022.

[25] 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.

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

[27] 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.

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

[29] 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.

[30] 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.

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

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

[33] Jie Lei et al. "Tvqa: Localized, compositional video question answering". In: *arXiv preprint arXiv:1809.01696* (2018).

[34] 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.

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

[36] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[37] Tegan Maharaj et al. "A Dataset and Exploration of Models for Understanding Video Data through Fill-in-the-Blank Question-Answering". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 7359–7368.

[38] Kenneth Marino et al. "Ok-vqa: A visual question answering benchmark requiring external knowledge". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3195–3204.

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

[40] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. "Recurrent models of visual attention". In: *Advances in neural information processing systems*. 2014, pp. 2204–2212.

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

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

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

[44] 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.

[45] 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.

[46] 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).

[47] 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*. Association for Computing Machinery, 2009.

[48] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[49] Gursimran Singh. “Spatio-temporal relational reasoning for video question answering”. PhD thesis. University of British Columbia, 2019.

[50] Robyn Speer, Joshua Chin, and Catherine Havasi. “Conceptnet 5.5: An open multilingual graph of general knowledge”. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[51] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[52] 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.

[53] Du Tran et al. “Learning spatiotemporal features with 3d convolutional networks”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4489–4497.

[54] Peng Wang et al. “Explicit Knowledge-based Reasoning for Visual Question Answering”. In: *IJCAI-17*. 2017, pp. 1290–1296.

[55] Peng Wang et al. “Fvqa: Fact-based visual question answering”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.10 (2018), pp. 2413–2427.

[56] Dejing Xu et al. "Video Question Answering via Gradually Refined Attention over Appearance and Motion". In: *MM '17*. 2017.

[57] Jun Xu et al. "MSR-VTT: A Large Video Description Dataset for Bridging Video and Language". In: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[58] Yunan Ye et al. "Video question answering via attribute-augmented attention network learning". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. 2017, pp. 829–832.

[59] Youngjae Yu et al. "End-to-End Concept Word Detection for Video Captioning, Retrieval, and Question Answering". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 3261–3269.

[60] 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, 4334–4340.

[61] Linchao Zhu et al. "Uncovering the temporal context for video question answering". In: *International Journal of Computer Vision* 124.3 (2017), pp. 409–421.

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 [17]. 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

$$sort\_name(c). \tag{A.1}$$

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

$$static(g). \tag{A.2}$$

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

$$fluent(inertial, f). \tag{A.3}$$

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

$$fluent(defined, f). \tag{A.4}$$

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

$$action(a). \tag{A.5}$$

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

$$\#const\ n = \ < max > . \tag{A.6}$$

$$step(0..n). \tag{A.7}$$

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

$$h(l, I+1) \text{ :- } h(p_0, I), ..., h(p_m, I), occurs(a, I), I < n. \tag{A.8}$$

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

$$h(l, I) \text{ :- } h(p_0, I), ..., h(p_m, I). \tag{A.9}$$

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

$$\neg occurs(a_0, I); ...; \neg occurs(a_k, I) \text{ :- } h(p_0, I), ..., h(p_m, I). \tag{A.10}$$

The ; in the head of rule A.10 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:

$$\begin{aligned} holds(F, I+1) \text{ :- } & fluent(inertial, F), \\ & holds(F, I), \\ & not\ \neg holds(F, I+1), \\ & I < n. \end{aligned} \tag{A.11}$$

$$\begin{aligned} \neg holds(F, I+1) \text{ :- } & fluent(inertial, F), \\ & \neg holds(F, I), \\ & not\ holds(F, I+1), \\ & I < n. \end{aligned} \tag{A.12}$$

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

$$\neg holds(F, I) \text{ :- } fluent(defined, F), step(I), not\ holds(F, I). \tag{A.13}$$

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

$$\neg occurs(A, I) \text{ :- } action(A), step(I), not\ occurs(A, I). \tag{A.14}$$

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 $\neg holds(F, I)$.