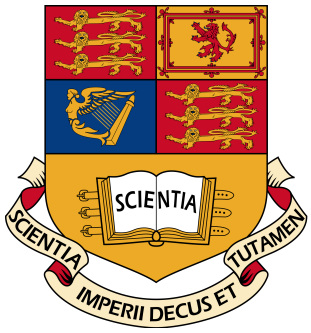

DEPARTMENT OF COMPUTING

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

H-PERL: The Hybrid Property, Event and Relation Learner

Author:
Ross Irwin

Supervisors:
Prof. Alessandra Russo
Dr. Krysia Broda
Dr. Jorge Lobo

June 4, 2020

Chapter 1

Hardcoded Model

Our first H-PERL implementation is the 'hardcoded' model. The hardcoded model makes use of a mixture of manually engineered components and components which are trained on the full-data version of the OceanQA dataset. This model should not be taken as a solution to the general VideoQA problem, since it would be labourious to rewrite components for each new dataset environment. Instead we intend this model to be used as a benchmark for the OceanQA dataset, against which other VideoQA implementations can be evaluated.

Full details on the performance of the model, as well as the performance of some of the individual components is given in Chapter **??**.

1.1 Properties

As mentioned in Chapter **??**, the role of the properties component is to take an image of an object (as produced by the object detector) and, in the case of the OceanQA dataset, to return the colour and rotation of the object.

Since the object image is a 3D tensor[1] of raw pixel values, a convolutional neural network is an excellent candidate for the implementation of the properties component. Other computer vision techniques for machine learning such as decision trees, random forests and SVMs require thousands of manually engineered filters to be applied to the image, whereas convolutional networks can learn a much smaller set of filters based on the training data.

Figure 1.1 shows the architecture of the properties network. The first part of this network encodes the object image into a 1024 dimenional latent vector using a series of convolutional and fully-connected layers. A set of fully-connected layers, one for each property, each take the vector encoding of the object and produce a vector of real numbers. The size of each vector is equal to the number of possible values that property can take.

[1] Height, width and RGB colour channels make up the three dimensions

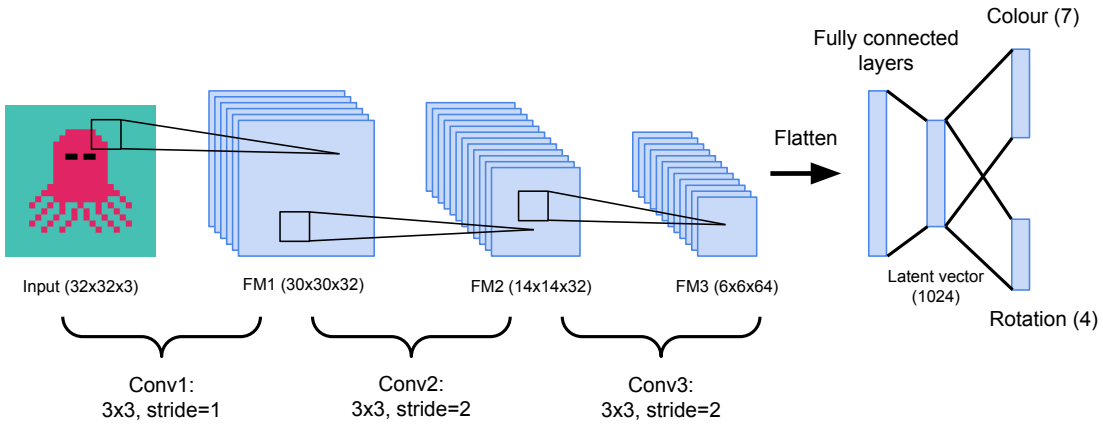

Figure 1.1: Architecture of the properties component neural network. An object is encoded into a latent vector, before a set of fully-connected layers produce a set of probability distributions over the property values. Batch normalisation is applied between convolutional layers. FM stands for feature maps.

The final output of the network is a set of probability distributions, one for each property, over the set of possible values that property can take. As is standard in a multiclass classification problem, the softmax function is applied to each set of property values in order to construct the probability distribution. The softmax function guarantees that the elements of a vector sum to one and that each element is greater than or equal to zero, hence softmax creates a discrete probability distribution. The softmax function for a vector $\mathbf{z} \in \mathbb{R}^K$ is as follows:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, ..., K \tag{1.1}$$

The full-data version of the OceanQA dataset labels both the colour and rotation of every object in every frame. This makes training a neural network relatively straightforward; we collate all of the objects and their properties in the dataset, resize each object to 32x32 pixels, convert each property into a one-hot encoded vector and train the network using batches of 256 objects with a learning rate of 0.001 for 2. The cross-entropy loss is calculated for each property and these are summed to give an overall loss. The cross-entropy loss between a predicted probability distribution vector $\mathbf{p} \in \mathbb{R}^K$ and a one-hot encoded classification vector $\mathbf{y} \in \mathbb{R}^K$, where K is the number of classes, is as follows:

$$H(\mathbf{p}, \mathbf{y}) = -\sum_{c=1}^{K} y_c \log(p_c) \tag{1.2}$$

When the H-PERL model is being evaluated each object in the video is applied to the network (batched together for efficiency) and a set of probability distributions is produced. For each object, the property value for a particular property is given by the most probable element of the vector.

1.2 Relations

The job of the relations component is to list all instances of relevant binary relations between objects in each frame of the video. Since there is only one relevant binary relation in the OceanQA dataset, the job of this component is simply to list the instances of the *close* relation.

The relations component of the hardcoded model contains a hand-written binary classification algorithm for determining whether two objects are close or not. For a given video, this algorithm is applied to every pair of objects in every frame of the video in order to list all instances of the relation. The object arguments of the closeness algorithm are given in symbolic form, rather than as raw pixel matrices. This means the algorithm is heavily reliant on accurate information from the object detector - the position tuple in particular. Although the closeness algorithm doesn't make use of the property component's extracted features, other algorithms for determining binary relations may require these.

The algorithm for determining the closeness of two objects is, in fact, identical to the algorithm used when constructing the dataset. The algorithm uses the idea of an expanded box around each object; the objects are close if their boxes overlap, as can be seen in Figure 1.2. The algorithm was discussed alongside techniques used to create the dataset in Chapter **??** and is shown in Algorithm **??**.

Clearly, since the algorithm used to construct the dataset and the algorithm used to find the relations between objects are exactly the same, the relations component achieves perfect accuracy provided the object detection is exactly correct. Although this may seem like cheating, since in general it is not possible to know the underlying rules of the dataset, we reiterate that the hardcoded model is not a solution to VideoQA tasks in general, but rather is specific to the OceanQA dataset, and can be used for comparisons with other models.

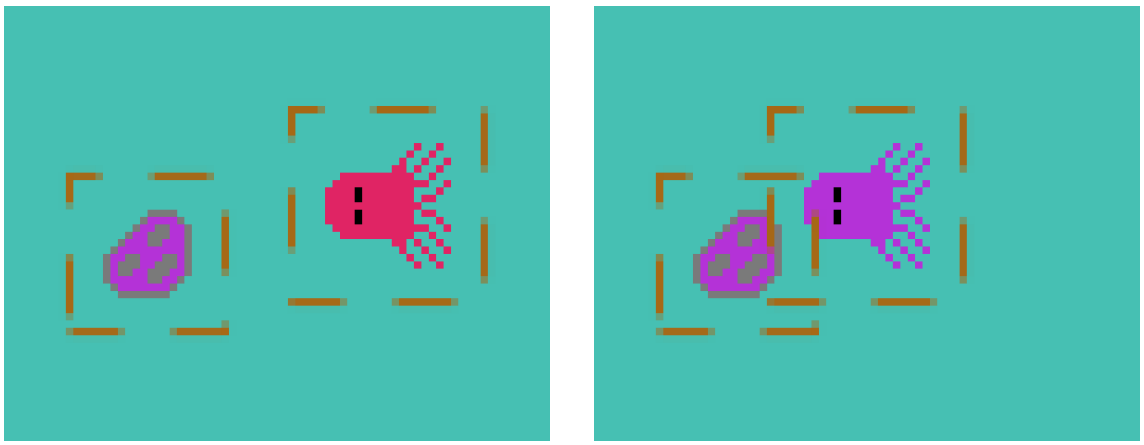

Figure 1.2: Diagrams showing the octopus before and after moving close to a purple rock, and therefore turning purple itself. The brown dashed lines show the bounding boxes expanded by 5 pixels on each side around the objects. If these boxes overlap the objects are deemed to be close to one another.

Another consideration for this approach to relation classification is speed; each relation classification algorithm (of which there is only one in the OceanQA dataset, but, in general there may be many) looks at every possible pair of objects in every frame of the video. Assuming that each relation classifier operates in constant time, this creates an overall algorithmic complexity of $\mathcal{O}(kmn^2)$, where k is the number of frames per video, m is the number of relations to be classified and n is the number of objects in each frame. Despite this we found that the time taken by the relation component was small relative to other components in the H-PERL model. Chapter **??** outlined the full details of the component's performance, including details on the time taken during evaluation.

1.3 Events

As mentioned in Chapter **??**, the role of the event detection component is to list all of the actions and effects which take place between each pair of consecutive frames of a given video. Since preceeding components have extracted a number of object and frame-level symbolic features already, the event detector in the hardcoded model works only with these features, as opposed to viewing the raw pixels in each frame.

For the hardcoded model, the event detector is implemented by, firstly, searching through all possible combinations of actions. The component assumes that there is only one action per frame and that only non-static objects can be the cause of an action. Each combination of actions is then applied to a hand-written $\mathcal{AL}$ model of the environment to generate the set of symbolic features which would be expected to be observed should the set of actions have occurred. This set of features is then compared to the set of features which were actually observed - the set of features extracted by preceeding components. Finally, the combination of actions which lead to the best fit with the observed data are selected.

For each set of possible actions and their associated features generated by the $\mathcal{AL}$ model, a set of manually engineered ASP rules is used to find the corresponding set of effects. For example, if the octopus moves[2] during frame i and is close to a blue rock during frame $i+1$, then we can infer that the *change colour* effect occurs during frame i.

1.3.1 Actions

1.3.2 Effects

1.4 Error Correction

[2]Moving during frame i refers to an object moving between frame i and frame $i+1$.