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. The final output of the network is a set of probability

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

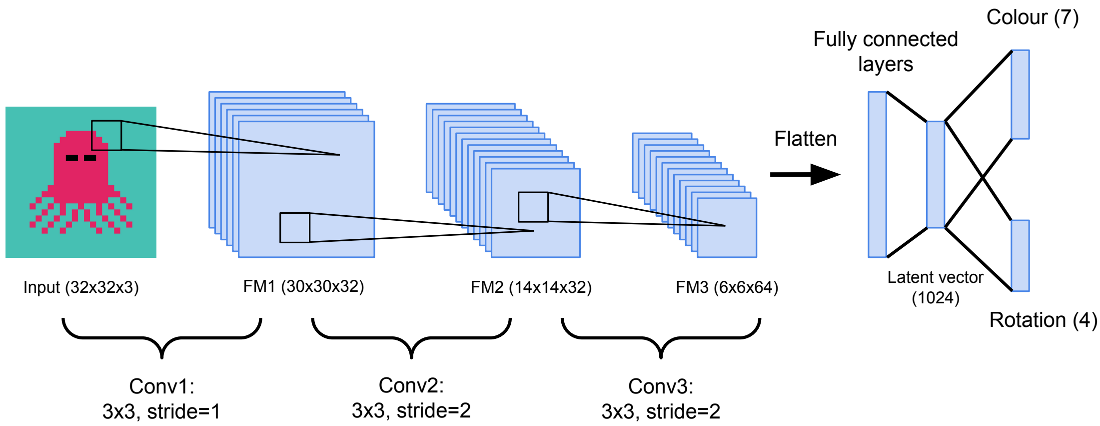

Figure 1.1: Architecture of the properties component neural network. An object is encoded into a latent vector, before a set of fully-connected layers produce a set of probability distributions over the property values. FM stands for feature maps.

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 simply collate all of the objects in the dataset, resize each object to 32x32 pixels and train the network using batches of 256 objects with a learning rate of 0.001 for 2 epochs.

1.2 Relations

1.3 Events

1.4 Error Correction