Technical report M2CAI Workflow Challenge Method 2: CNN and time smoothing

Remi Cadene, Thomas Robert, Nicolas Thome, and Matthieu Cord

Universit Pierre et Marie Curie, UPMC-Sorbonne Universits, LIP6, Paris, France

Abstract. This report present our work on the M2CAI workflow challenge consisting in recognizing at which step of an operation each frame of a video belong. This was done using a deep convolutional neural network followed by a step of temporal smoothing using simple temporal averaging of the predictions and an hidden Markov model of the process. **Please note that this method is a derivative of our method 0 adding an HMM to improve predictions**

1 Introduction

The *M2CAI workflow* challenge consists in analyzing endoscopic videos of minimally invasive surgical operations of cholecystectomy. The task at hand is to detect at which of the 8 phases of the operation each frame belong. This can be useful to evaluate a surgeon or to trigger automatic actions in the operating room for example.

For this challenge, the task must be carried out "online", meaning the prediction for a given frame can only be made based on past frames and predictions, without any knowledge of the future. This simulates a process of prediction in real-time.

The dataset consists of 27 videos in the training set, ranging from 14 to 66 minutes and a test set of 15 videos. The videos have a resolution of 1920 × 1080 pixels and are shot at 25 frames per second at the IRCAD research center in Strasbourg, France.

1.1 Our approach

The approach described in this paper consists of two main step, one step of visual recognition without any temporal component using a deep Convolutional Neural Network (CNN), and one step of temporal smoothing to improve the coherence of the predicted sequence using averaging and a Hidden Markov Model (HMM).

2 Visual recognition: Deep CNN without validation set

The first step of our approach is to analyze each step of the video.

For this, first we extract one frame every 25 frames, returning 1 frame per second of the video. This is done both on the train and test sets.

We split the train data into two sets, a training set of 22 videos and a validation set of 5 videos.

We train an Inception-V3 [2] model and validating the different hyperparamaters for the optimization. We keep the weights at the epoch that gave the best results on the validation set.

Note that the network is initialized with the weights learnt for the ILSVRC challenge (classification of 1.2 million images across 1000 categories). It is fine tuned for this task with Adam [1] optimization algorithm.

3 Temporal smoothing: averaging

3.1 Averaging

From the previous step of our method, we obtain for each image in the test set a vector of log-probabilities, with for each possible step the number of networks predicting that the frame belong to this step of the operation.

To temporally smooth the predictions, we average the vectors of log-probabilities across the last 15 frames (corresponding to 15 seconds of the video). This means that our smoothed prediction has a lag of 7.5 seconds. However, the metric of the challenge allows a 10-second-margin in the predictions, meaning that it is not considered problematic to have a slight lag in our prediction. We therefore take advantage of this tolerance to improve the smoothness of the predictions.

3.2 Hidden Markov Model

In addition to the averaging, we propose to use an HMM to model the transition between the various steps of the operation.

To do so, we consider that the step of the operation is the discrete hidden state x_t, from which we only observe a noisy vector y_t that is our averaged vector of log-probabilities from the network.

Training An HMM has 3 parameters: the initial state probabilities, the matrix of probabilities of transition between states from one timestep to the next, and the parameters regarding the emission of the observations for each possible step.

Those information are directly accessible in the trainset and computed on it. Probabilities are computed by simple counting. We chose to model the emission of observation with a gaussian distribution so for each step we compute the average observation and its covariance matrix.

Offline prediction An HMM is especially useful in offline mode, especially in our case where the transition matrix is sparse imposing a lot of structure on the predicted sequence of states. By applying Viterbi algorithm, we can “decode” a sequence of observations (votes from the networks) to obtain the most likely sequence of states. This is done for our offline prediction.

Online prediction However, for this challenge, the predictions must be given in online mode. Therefore, for our online predictions, to predict the state x_t we apply the Viterbi algorithm on the sequence $y_1, ..., y_t$ and keep the last state of the predicted sequence. This process ensure we are working in only mode but decrease the performance of the HMM, because the constrains on the transition matrix are no longer really enforced on the predicted sequence.

4 Post-processing to produce requested files

Our previous step gives us prediction at a rate of 1 frame per second. To come back to the required rate of 25 fps, we simply copy each prediction 25 times, and then crop or pad the predicted sequence to match exactly the number of frames in the video.

5 References

1. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. ICLR (2014)
2. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. CVPR (2015)