M2CAI Workflow Challenge : Convolutional Neural Networks with Time Smoothing and Hidden Markov Model for Video Frames Classification

Remi Cadene Thomas Robert Nicolas Thome Matthieu Cord
Sorbonne Universites, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu, 75005 Paris
{remi.cadene, thomas.robert, nicolas.thome, matthieu.cord@lip6.fr}

Abstract

Our approach is among the three best to tackle the M2CAI Workflow challenge. The latter consists of recognizing the operation phase for each frames of endoscopic videos. In this technical report, we compare several models with our best solution, namely a fine tuned Residual Network-200 on 80% of the training set with temporal smoothing using simple temporal averaging of the predictions and a Hidden Markov Model modeling the transitions.

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 frames 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.

Our experiments can be reproduce using our code : `http://github.com/Cadene/torchnet-m2caiworkflow`. Thus, we will not delve into too much details about our implementation.

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

The first step of our approach is to learn a classification model of video frames.

To do so, we split randomly the training data into two sets of videos, a full training set of 22 videos and a validation set of 5 videos. In our approach, the training set is made of the following video : 1,3,4,5,6,7,8,11,12,14,15,16,17,18,19,20,21,22,23,24,25,26. The testing set is made of : 2,9,10,13,27.

Secondly, we extract one frame every 25 frames, returning 1 frame per second of the video. This is done both on the full training set and testing set (the one without any label).

Thirdly, we train our frame classifier and validate it on our validation set. In this study, we compare several approaches detailed in the following subsections.

2.1. Pretrain CNN as Features Extractor with SVM

This approach consists of using a pretrain CNN to vectorize the training and validation sets and to train a multi-class classifier on the features vectors, such as a linear model with a cross entropy criterion or Support Vector Machines with a one versus all strategy. Usually the features are extracted at the end of the CNN after a fully connected layer or an activation functions. This is the usual baseline method for transfer learning

In this challenge, we use features extracted from the penultimate layer of InceptionV3 [?] (2048 dimensions) and train a linear model with a cross entropy criterion. Our preprocessing is as follow. As InceptionV3 takes as input images of size 299x299. We randomly rescale the original images between 299 and 330 pixels of width or height based on the smallest fitting dimension. Then, we randomly crop a square region of size 299x299. Finally, we normalize each pixels using the mean and standard deviation processed on ImageNet.

2.2. Fine Tuning Pretrain CNN

Fine tuning consists of training a pretrain CNN on a smaller dataset. Typically, the last fully connected layers, which can be viewed as classification layers, are reset and a smaller learning rate is applied to the pretrain layers. By doing so, the goal is to adapt the representations learned to the new dataset. The more different is the latter from the original dataset, the more layers must be reset.

In this study, we remove the last layer of a pretrain InceptionV3 on ImageNet and add a new fully connected of output size equal to 4. We use Adam [**?**] which does not need any smaller learning rate for pretrain layers. We tune using grid search the learning rate et the learning rate decay (which decrease the adaptive learning rate of each parameters by a factor after each mini batch).

We also fine tune an other pretrain CNN on ImageNet called Residual Network-200. As it takes as input images of size 221x221, we use the same preprocessing with a minimal size and a crop size of 221 and a maximum size of 240.

2.3. Other Baselines

The first baseline is to train an InceptionV3 From Scratch. It means that we reset all its parameters with LSUV method

The second baseline is to Fine Tune an Inception-V3 with the WELDON aggregation layer plugged at the end. We use the same preprocessing with a minimal size and crop size of 448 and a maximum size of 463.

3. Temporal smoothing: Averaging and HMM

3.1. Averaging

Using a model defined in the previous section, we obtain for each images a vector of log-probabilities. To temporally smooth the predictions, we average the vectors 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 transitions 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 A HMM has 3 kind of parameters: the initial state probabilities, the matrix of probabilities of transition between states from one time-step to the next, and the parameters regarding emission of observations for each possible step.

Classification Model	Validation Accuracy Top 1 (%)
InceptionV3 From Scratch	69.13
IncpetionV3 Weldon	78.18
InceptionV3 Extraction	78.26
InceptionV3 Fine Tuned	79.06
ResNet200 Fine Tuned	79.24

Table 1.

Temporal Model	Score (%)
Baseline	
Offline	
Online	

Table 2.

We compute those information on the training set. The initial state probabilities and the matrix of transition are computed by simple counting. We chose to model the emission of observation with a gaussian distribution. To do so, we computer for each step the average observation and its covariance matrix.

Offline prediction In this study, the transition matrix of our trained HMM is parse, which impose a lot of structure on the predicted sequence of states. By applying Viterbi algorithm, we “decode” a sequence of observations 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, 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 that we are working in online mode. However, it decreases the performance of the HMM, because the constrains on the transition matrix are no longer enforced on the predicted sequence.

3.3. Post-processing to produce requested files

As for now, our two-step model gives us predictions 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.

4. Experiments

VIsualisation

5. Conclusion

In this challenge, we tried several approaches. We validated that fine tuning Convolutional Neural Networks is a good approach on this kind of datasets.

References