Artificial Vision and Pattern Recognition project

Salvatore Davide Amodio
id number : CA29166EP

Recognizing Human Actions in Still Images using Machine and Deep Learning Models

The purpose of this task was to develop two systems for predicting actions from still images. The former required to train an SVM with the features extracted during a pre-processing step while the latter relied on pre-trained deep learning networks. The dataset is composed of all daily human actions images concerning seven different actions : *Interacting with the computer*, *Photographing*, *Playing an Instrument*, *Riding a Bike*, *Riding a Horse*, *Running*, and *Walking*.

First system with a pre-processing stage and based on SVM After a pre-processing stage, which consists first in smoothing and resizing the images (256×256) and then in applying histogram equalization for normalizing illumination effects, a traditional method called Local Binary Pattern (LBP) has been used to extract features from action images. An exhaustive search has been carried out by implementing a grid search to find the best hyper-parameters for the task under consideration.

$cellSize$	6	10	20	30	40
	50	60	70	80	90
	100	110	120	130	
SMV_Kernel	poly	linear	rbf		
SVM_C	0.001	0.05	0.01	0.5	0.1
	1	3			

`extractLBPFeatures` method allows to set a parameter called CellSize used for defying the number of cells, calculated as $floor(size(I)/CellSize)$. SVM_Kernel and SVM_C are hyper-parameters of `fitcecoc` method where the former defines the type of kernel function used in the SVM while the latter adds a penalty for each misclassified data point. In a nutshell if C is small, the penalty for misclassified points is low; so a decision boundary with a large margin is chosen at the expense of a greater number of misclassifications. There is another hyper-parameter called *kernelScale* that is usually tuned like the previous hyper-parameters, but it has been decided to fix it on $'auto'$; an

appropriate scale factor has been selected using a heuristic procedure by the `fitcecoc` method. So, for each values combination, a machine learning model has been trained on the `trainingSet` and validated on the `validationSet`.

$$|cellSize| \times |SMV\_Kernel| \times |SVM\_C| = 294$$

Among the 294 machine learning models trained, the best fit model (the model that achieved the highest accuracy value on the validation set) for this task has those hyper-parameters and accuracy:

$cellSize$	60
SMV_Kernel	rbf
SVM_C	5
$TrainingAccuracy$	100%
$ValidationAccuracy$	42.54%

Once the model has been chosen and a classifier obtained, the performances have been estimated on the `testSet`. The proposed model got a level of accuracy on the test set of 40.22%.

Second system based on pre-trained deep learning network For the second system an end-to-end model based on a pre-trained deep learning network has been built. To provide transfer learning, for each task at least the most important characteristics must be considered for choosing the network to be applied: network accuracy, speed, and size.

GoogleNet has been chosen among several networks available because it has a good accuracy by considering the relative prediction time using GPU. Since the machine on which the second system has been trained did not have a GPU, it has been decided to use a lightweight neural network in order to achieve a good performance in a short time. In addition to replace the last three layers of GoogleNet (it has been necessary for adapting the network to the new task), another main architectural choice was to freeze the first 110 layers of the network, thus the weights contained within them did not change during the training. This practice sped up training and it is recommended when the new data set is small; freezing earlier network layers

can also prevent those layers from overfitting to the new data set. After the training, the classifier achieved these levels of accuracy:

$TrainingAccuracy$	98.12%
$ValidationAccuracy$	84.29%

In conclusion, on the test set the accuracy was of 82.86%.

Conclusion Taking into account only the level of accuracy provided by the two different systems on the test set, the latter is decisively better. Overall, the results attained by the second system have been surprising by considering the low number of epochs; only 6 which is very impressive. This reveals how powerful is the transfer learning into an image classification task like this one that has been dealt tackled. One reason why the first system did not get acceptable performance might be caused by noisy data where target classes overlap; that happens when the features have very similar or overlapping properties.