# Video frame classification for sport motion detection

Simone Cremasco VR468971

Master Degree in Computer Engineering for Robotics and Smart Industry, Academic Year 2020/2021

Machine Learning Course Project

## Contents

# 1 Motivation and Rationale

The idea for this project comes from a personal experience. During this winter I went to the mountains and due to Covid-19 situation, skilifts were close so the only option was to go up hill by walking. So I found a spot where I could "easily" walk up and built a small ramp from where to jump with a snowboard. Aside to that ramp I've placed a sport cam to record jumps which was placed steady and kept recording. At the end of day I had 2 hours of video but only a small portion of that was worth been watched. In fact each jump last only a couple of seconds while climbing up and recover for the hard walk takes a lot of time. Here's where the idea came from: for these kind of situations it would be handy having a tool that automatically recognise when you are doing some activity in a video record and cut out those scene for you.

# 2 State of the art

As I'm tackling the problem as a classification problem to find a specific content in a video, the main model I'm using is a Convolutional Neural Network (CNN), because of the high accuracy it can provide. Moreover the idea is to apply transfer learning: when comes to image classification there are already pre-trained networks with millions of images. On such networks that weights are already settled in a manner that early layers are able to extract a lot of features on images. Later layers can instead be readjusted in order to classify a small set of classes that don't need to belong to the original network. One approach can be to simply use the net as is (keeping the weights) to use it as a feature extractor, replacing the last layer with a linear classifier. This is helpful when we have a small dataset with similar images of original network as it will have enough features extracted for them. Another approach is to fine tune the network which means replacing the last layer as for the other approach but we also continue to back-propagate during training so that weights will be adjusted. This is particularly helpful when we have a large dataset as we can be more confident that adjusting weights won't result in overfitting but just shifts the context to the new classes.

# 3 Objective

The detection as described in motivation section can be achieved with methods that does not require the use of Machine Learning, using for example some computer vision techniques. For this reason the objective of this project is a little bit different from pure detection, it will be to classify which sport is depicted in each video frames. Using a confidence on classification I can ask the tool to show me at which moments during the video a particular sport is shown. In order to do that it is necessary to set some parameters before starting the process: the number of useful frames per second to be classified and a threshold above which we are confident that a sport is depicted. This tool should help

in cases of video taken from static pose, in which there's not always action in progress, with the idea to strip off only the action and not the static background. Finally we can feed the tool with the video, the sport we are looking for, the confidence and optionally the number of frames per second we want to check. About the number of frames I've set a default value of 10 as I think they should be enough for the average motion to get at least one action frame. This number should be instead tuned if for example we are recording something extremely fast with a 60 fps camera, so we can ask to check more frames each second.

# 4 Methodology

## 4.1 Dataset

The dataset [1] contains 73 different classes, each referring to a different sport, for a total of 10416 train images and 365 test images. Training images are more or less balanced, each class can have between 120 and 190 images to train on. Instead the testing images are exactly 5 for each class. Initially I've used the complete set of classes, but in the end I've decided to reduce the number of classes down to 18. Without this downsampling each train session (and PCA computation) would have taken many hours. Also I've doubled the test images (10 per classes) using the validation ones already included in dataset. The set has now 2742 training images and 180 test images and includes the following classes: baseball, basketball, bmx, football, formula 1 racing, frisbee, golf, hockey, motorcycle racing, rugby, skydiving, snow boarding, sumo wrestling, surfing, swimming, table tennis, tennis, volleyball.

## 4.2 Models

Starting from a pre-trained CNN based on imagenet (ResNet18) I've applied transfer learning on the dataset with two different approaches:

- fine tune (which we later refer as FT), changing the last layer and let the entire network readjust the weights;
- feature extractor (which we later refer as FE), changing the last layer and train only on that one using previous layers as fixed.

For comparison I've also used:

- One vs One Support Vector Classifier (which we later refer as SVC_OVO), with 'rbf' kernel;
- One vs Rest Support Vector Classifier (which we later refer as SVC_OVR), with 'rbf' kernel;
- K Nearest Neighbours Classifier (which we later refer as KNN), with 5 neighbours.

For KNN, k=5 have been selected after checking the best test accuracy among 30 different values of k, as we can see on (fig. 1).

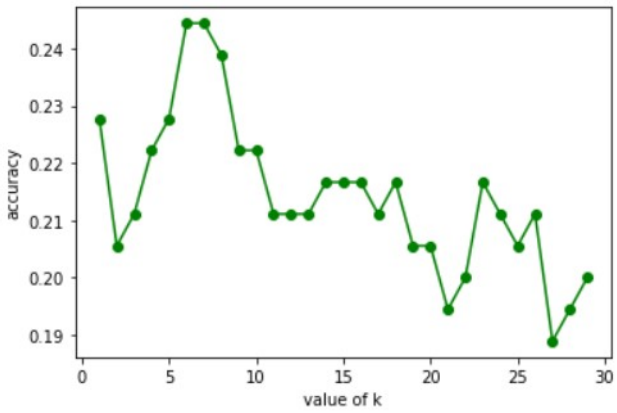



Figure 1: Knn accuracy over differtent k values, 1 to 30

### 4.3 Implementation

The transfer learning approach comes from pytorch tutorial [2], studied and adapted to the project. The validation metric is the best accuracy for test set. A consideration is needed about model output: during training the linear layer is enough to get the maximum value but in order to get the confidence during real cases I've used a Softmax function on the output. The other 3 models are instances of scikit learn models; for them the data has been scaled with StandardScaler and transformed using PCA with 80% explained variance. The result is 189 (fig. 2) features instead 50176 (224*224 images). The "frame

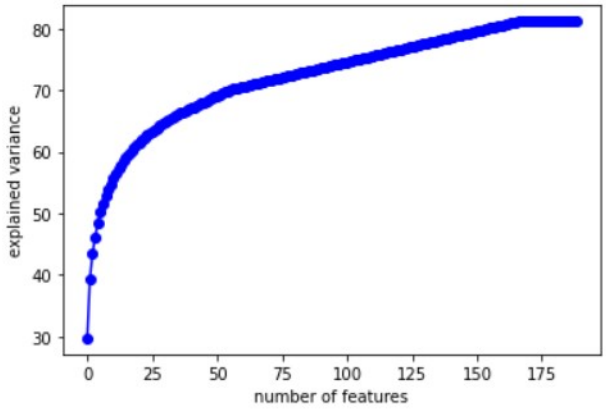



Figure 2: PCA explained variance up to 80%

extractor" script code uses some OpenCv functions to extract frames, extract some information such as FPS and apply some editing on images. About the

analysis: metrics from scikit learn have been used, applying a weighted average on precision and recall scores to have a more fair result.

# 5 Experiments and Result

As said before, I've started with all the classes and trained the CNNs with those where both networks were trained for 25 epochs on GPU and it took more than 2 hours for fine tune model and almost 1 hour for feature extractor model. I will now show those results, and then show a fair result with all 5 methods trained on just 18 classes. Now CNNs took respectively 28 minutes and 11 minutes, trained on 30 epochs. We can see (fig. 3) that at around 8 epochs accuracy and loss settled for both CNNs.

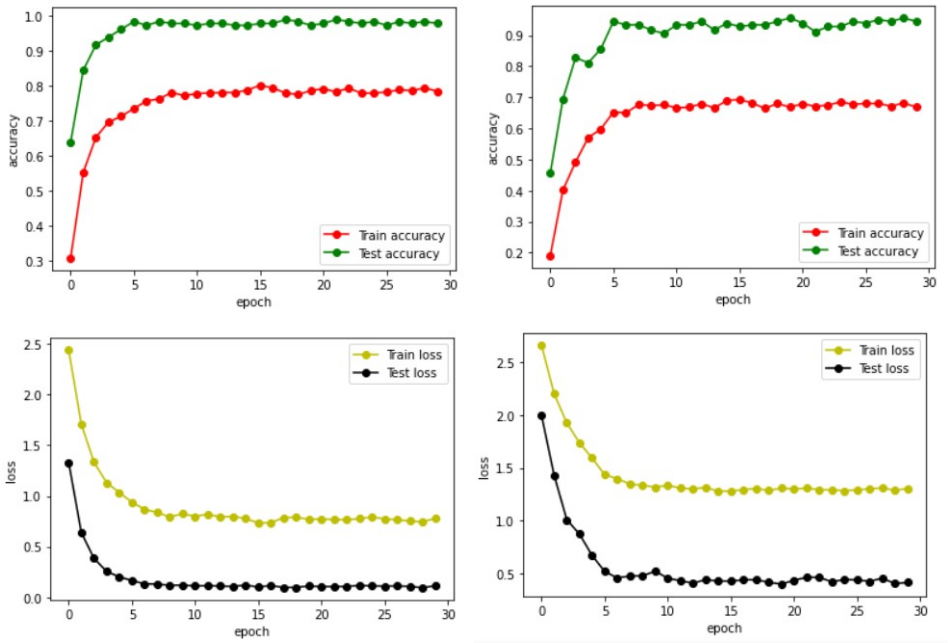



Figure 3: Loss and accuracy during training epochs. Ft model on the left, FE model on the right

### 5.1 Analysis of models

| CNNs trained on 73 classes | | | |
|---|---|---|---|
| Model and set | Accuracy | Precision | Recall |
| FT on Test set | 0.942 | 0.949 | 0.942 |
| FT on Train set | 0.832 | 0.838 | 0.832 |
| FE on Test set | 0.942 | 0.950 | 0.942 |
| FE on Train set | 0.795 | 0.801 | 0.795 |

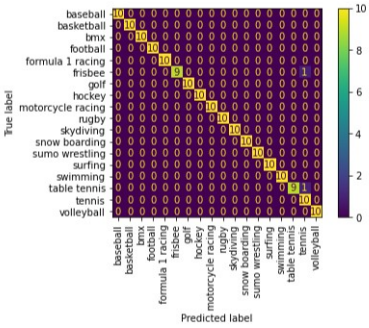
| All models trained on 18 classes | | | |
|---|---|---|---|
| Model and set | Accuracy | Precision | Recall |
| FT on Test set | 0.989 | 0.990 | 0.989 |
| FT on Train set | 0.896 | 0.899 | 0.896 |
| FE on Test set | 0.955 | 0.962 | 0.955 |
| FE on Train set | 0.814 | 0.815 | 0.814 |
| SVC_OVO on Test set | 0.389 | 0.418 | 0.389 |
| SVC_OVO on Train set | 0.611 | 0.641 | 0.611 |
| SVC_OVR on Test set | 0.389 | 0.419 | 0.389 |
| SVC_OVR on Train set | 0.616 | 0.637 | 0.616 |
| KNN on Test set | 0.244 | 0.329 | 0.244 |
| KNN on Train set | 0.319 | 0.484 | 0.319 |

What matter most for the purpose of the project is the recall score, in fact we can accept some false positives (just some, otherwise it becomes useless) as the user can eventually ignore them but we want that all real positives will be found, so no false negative (high recall) to not lose any recorded action. Highest recall is underlined in the table above and belongs to FT model.

In figures 4, 5, 6, 7, 8 we can see the analysis for every class and every model.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| baseball | 1.000 | 1.000 | 1.000 | 10 |
| basketball | 1.000 | 1.000 | 1.000 | 10 |
| bmx | 1.000 | 1.000 | 1.000 | 10 |
| football | 1.000 | 1.000 | 1.000 | 10 |
| formula 1 racing | 1.000 | 1.000 | 1.000 | 10 |
| frisbee | 1.000 | 0.900 | 0.947 | 10 |
| golf | 1.000 | 1.000 | 1.000 | 10 |
| hockey | 1.000 | 1.000 | 1.000 | 10 |
| motorcycle racing | 1.000 | 1.000 | 1.000 | 10 |
| rugby | 1.000 | 1.000 | 1.000 | 10 |
| skydiving | 1.000 | 1.000 | 1.000 | 10 |
| snow boarding | 1.000 | 1.000 | 1.000 | 10 |
| sumo wrestling | 1.000 | 1.000 | 1.000 | 10 |
| surfing | 1.000 | 1.000 | 1.000 | 10 |
| swimming | 1.000 | 1.000 | 1.000 | 10 |
| table tennis | 1.000 | 0.900 | 0.947 | 10 |
| tennis | 0.833 | 1.000 | 0.909 | 10 |
| volleyball | 1.000 | 1.000 | 1.000 | 10 |
| accuracy | | | 0.989 | 180 |
| macro avg | 0.991 | 0.989 | 0.989 | 180 |
| weighted avg | 0.991 | 0.989 | 0.989 | 180 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| baseball | 0.901 | 0.839 | 0.869 | 174 |
| basketball | 0.802 | 0.864 | 0.832 | 169 |
| bmx | 0.889 | 0.914 | 0.901 | 140 |
| football | 0.910 | 0.848 | 0.878 | 191 |
| formula 1 racing | 0.959 | 0.979 | 0.969 | 191 |
| frisbee | 0.791 | 0.880 | 0.833 | 133 |
| golf | 0.906 | 0.796 | 0.847 | 157 |
| hockey | 0.879 | 0.971 | 0.923 | 172 |
| motorcycle racing | 0.890 | 0.917 | 0.903 | 132 |
| rugby | 0.919 | 0.892 | 0.905 | 166 |
| skydiving | 0.972 | 0.946 | 0.959 | 149 |
| snow boarding | 0.926 | 0.855 | 0.889 | 117 |
| sumo wrestling | 0.939 | 0.932 | 0.936 | 133 |
| surfing | 0.978 | 0.944 | 0.961 | 142 |
| swimming | 0.974 | 0.949 | 0.961 | 157 |
| table tennis | 0.826 | 0.908 | 0.865 | 141 |
| tennis | 0.797 | 0.901 | 0.846 | 131 |
| volleyball | 0.900 | 0.796 | 0.845 | 147 |
| accuracy | | | 0.896 | 2742 |
| macro avg | 0.898 | 0.896 | 0.896 | 2742 |
| weighted avg | 0.899 | 0.896 | 0.897 | 2742 |



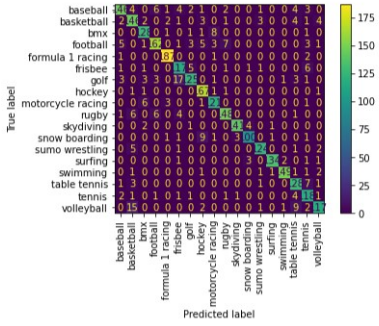
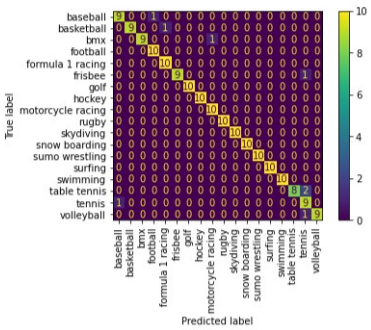




Figure 4: Analysis and Confusion matrix of FT model, test on the left, train on the right

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| baseball | 0.900 | 0.900 | 0.900 | 10 |
| basketball | 1.000 | 0.900 | 0.947 | 10 |
| bmx | 1.000 | 0.900 | 0.947 | 10 |
| football | 0.909 | 1.000 | 0.952 | 10 |
| formula 1 racing | 0.909 | 1.000 | 0.952 | 10 |
| frisbee | 1.000 | 0.900 | 0.947 | 10 |
| golf | 1.000 | 1.000 | 1.000 | 10 |
| hockey | 1.000 | 1.000 | 1.000 | 10 |
| motorcycle racing | 0.909 | 1.000 | 0.952 | 10 |
| rugby | 1.000 | 1.000 | 1.000 | 10 |
| skydiving | 1.000 | 1.000 | 1.000 | 10 |
| snow boarding | 1.000 | 1.000 | 1.000 | 10 |
| sumo wrestling | 1.000 | 1.000 | 1.000 | 10 |
| surfing | 1.000 | 1.000 | 1.000 | 10 |
| swimming | 1.000 | 1.000 | 1.000 | 10 |
| table tennis | 1.000 | 0.800 | 0.889 | 10 |
| tennis | 0.692 | 0.900 | 0.783 | 10 |
| volleyball | 1.000 | 0.900 | 0.947 | 10 |
| accuracy | | | 0.956 | 180 |
| macro avg | 0.962 | 0.956 | 0.957 | 180 |
| weighted avg | 0.962 | 0.956 | 0.957 | 180 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| baseball | 0.733 | 0.724 | 0.728 | 174 |
| basketball | 0.762 | 0.680 | 0.719 | 169 |
| bmx | 0.847 | 0.829 | 0.838 | 140 |
| football | 0.760 | 0.780 | 0.770 | 191 |
| formula 1 racing | 0.839 | 0.958 | 0.895 | 191 |
| frisbee | 0.762 | 0.722 | 0.741 | 133 |
| golf | 0.783 | 0.713 | 0.747 | 157 |
| hockey | 0.830 | 0.907 | 0.867 | 172 |
| motorcycle racing | 0.855 | 0.848 | 0.852 | 132 |
| rugby | 0.815 | 0.825 | 0.820 | 166 |
| skydiving | 0.820 | 0.946 | 0.879 | 149 |
| snow boarding | 0.893 | 0.641 | 0.746 | 117 |
| sumo wrestling | 0.843 | 0.887 | 0.864 | 133 |
| surfing | 0.802 | 0.944 | 0.867 | 142 |
| swimming | 0.895 | 0.866 | 0.880 | 157 |
| table tennis | 0.807 | 0.858 | 0.832 | 141 |
| tennis | 0.810 | 0.748 | 0.778 | 131 |
| volleyball | 0.857 | 0.735 | 0.791 | 147 |
| accuracy | | | 0.814 | 2742 |
| macro avg | 0.817 | 0.812 | 0.812 | 2742 |
| weighted avg | 0.815 | 0.814 | 0.812 | 2742 |



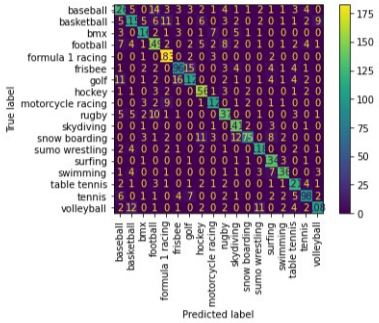




Figure 5: Analysis and Confusion matrix of FE model, test on the left, train on the right

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| baseball | 0.200 | 0.200 | 0.200 | 10 |
| basketball | 0.273 | 0.300 | 0.286 | 10 |
| bmx | 0.500 | 0.200 | 0.286 | 10 |
| football | 0.188 | 0.300 | 0.231 | 10 |
| formula 1 racing | 0.538 | 0.700 | 0.609 | 10 |
| frisbee | 0.500 | 0.100 | 0.167 | 10 |
| golf | 0.600 | 0.300 | 0.400 | 10 |
| hockey | 0.625 | 1.000 | 0.769 | 10 |
| motorcycle racing | 0.571 | 0.400 | 0.471 | 10 |
| rugby | 0.200 | 0.300 | 0.240 | 10 |
| skydiving | 0.455 | 0.500 | 0.476 | 10 |
| snow boarding | 0.500 | 0.300 | 0.375 | 10 |
| sumo wrestling | 0.444 | 0.400 | 0.421 | 10 |
| surfing | 0.545 | 0.600 | 0.571 | 10 |
| swimming | 0.280 | 0.700 | 0.400 | 10 |
| table tennis | 0.444 | 0.400 | 0.421 | 10 |
| tennis | 0.167 | 0.100 | 0.125 | 10 |
| volleyball | 0.500 | 0.200 | 0.286 | 10 |
| accuracy | | | 0.389 | 180 |
| macro avg | 0.418 | 0.389 | 0.374 | 180 |
| weighted avg | 0.418 | 0.389 | 0.374 | 180 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| baseball | 0.503 | 0.552 | 0.526 | 174 |
| basketball | 0.622 | 0.740 | 0.676 | 169 |
| bmx | 0.673 | 0.529 | 0.592 | 140 |
| football | 0.536 | 0.628 | 0.578 | 191 |
| formula 1 racing | 0.611 | 0.848 | 0.711 | 191 |
| frisbee | 0.941 | 0.241 | 0.383 | 133 |
| golf | 0.795 | 0.420 | 0.550 | 157 |
| hockey | 0.761 | 0.890 | 0.820 | 172 |
| motorcycle racing | 0.782 | 0.598 | 0.678 | 132 |
| rugby | 0.519 | 0.657 | 0.580 | 166 |
| skydiving | 0.585 | 0.530 | 0.556 | 149 |
| snow boarding | 0.709 | 0.624 | 0.664 | 117 |
| sumo wrestling | 0.736 | 0.714 | 0.725 | 133 |
| surfing | 0.638 | 0.669 | 0.653 | 142 |
| swimming | 0.455 | 0.764 | 0.570 | 157 |
| table tennis | 0.519 | 0.567 | 0.542 | 141 |
| tennis | 0.514 | 0.420 | 0.462 | 131 |
| volleyball | 0.765 | 0.422 | 0.544 | 147 |
| accuracy | | | 0.611 | 2742 |
| macro avg | 0.648 | 0.601 | 0.601 | 2742 |
| weighted avg | 0.642 | 0.611 | 0.604 | 2742 |

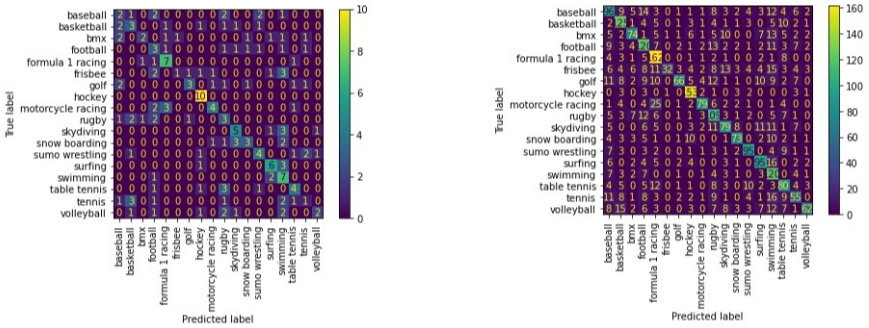
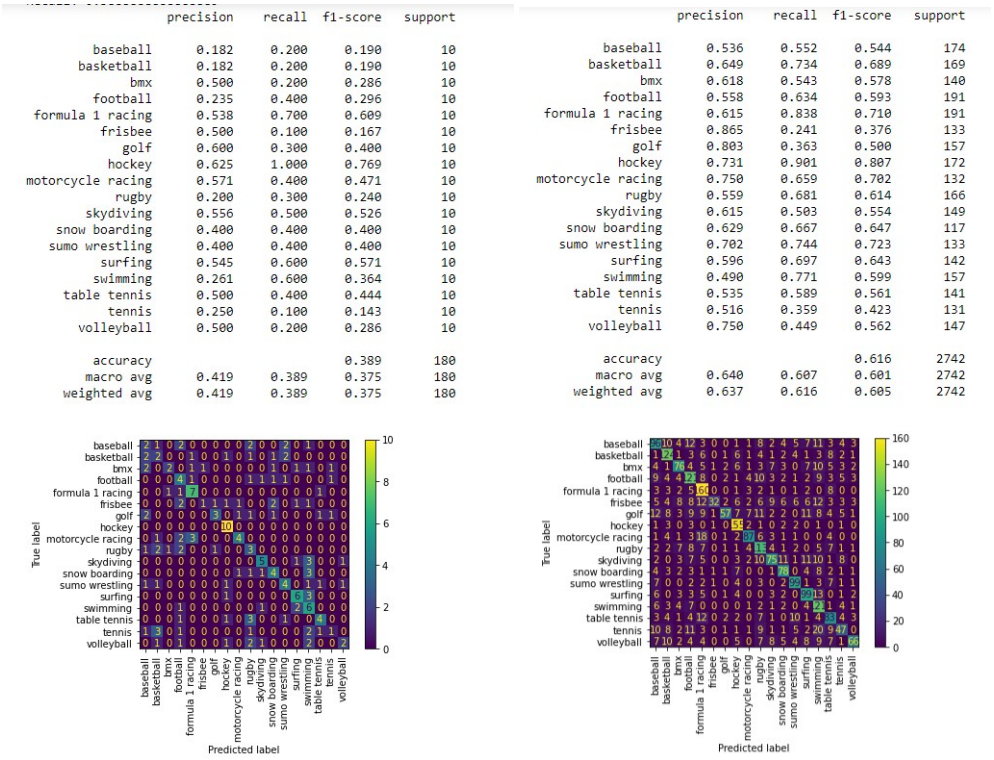



Figure 6: Analysis and Confusion matrix of SVC_OVO model, test on the left, train on the right



| | precision | recall | f1-score | support |
|---|---|---|---|---|
| baseball | 0.182 | 0.200 | 0.190 | 10 |
| basketball | 0.182 | 0.200 | 0.190 | 10 |
| bmx | 0.500 | 0.200 | 0.286 | 10 |
| football | 0.235 | 0.400 | 0.296 | 10 |
| formula 1 racing | 0.538 | 0.700 | 0.609 | 10 |
| frisbee | 0.500 | 0.100 | 0.167 | 10 |
| golf | 0.600 | 0.300 | 0.400 | 10 |
| hockey | 0.625 | 1.000 | 0.769 | 10 |
| motorcycle racing | 0.571 | 0.400 | 0.471 | 10 |
| rugby | 0.200 | 0.300 | 0.240 | 10 |
| skydiving | 0.556 | 0.500 | 0.526 | 10 |
| snow boarding | 0.400 | 0.400 | 0.400 | 10 |
| sumo wrestling | 0.400 | 0.400 | 0.400 | 10 |
| surfing | 0.545 | 0.600 | 0.571 | 10 |
| swimming | 0.261 | 0.600 | 0.364 | 10 |
| table tennis | 0.500 | 0.400 | 0.444 | 10 |
| tennis | 0.250 | 0.100 | 0.143 | 10 |
| volleyball | 0.500 | 0.200 | 0.286 | 10 |
| accuracy | | | 0.389 | 180 |
| macro avg | 0.419 | 0.389 | 0.375 | 180 |
| weighted avg | 0.419 | 0.389 | 0.375 | 180 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| baseball | 0.536 | 0.552 | 0.544 | 174 |
| basketball | 0.649 | 0.734 | 0.689 | 169 |
| bmx | 0.618 | 0.543 | 0.578 | 140 |
| football | 0.558 | 0.634 | 0.593 | 191 |
| formula 1 racing | 0.615 | 0.838 | 0.710 | 191 |
| frisbee | 0.865 | 0.241 | 0.376 | 133 |
| golf | 0.803 | 0.363 | 0.500 | 157 |
| hockey | 0.731 | 0.901 | 0.807 | 172 |
| motorcycle racing | 0.750 | 0.659 | 0.702 | 132 |
| rugby | 0.559 | 0.681 | 0.614 | 166 |
| skydiving | 0.615 | 0.503 | 0.554 | 149 |
| snow boarding | 0.629 | 0.667 | 0.647 | 117 |
| sumo wrestling | 0.702 | 0.744 | 0.723 | 133 |
| surfing | 0.596 | 0.697 | 0.643 | 142 |
| swimming | 0.490 | 0.771 | 0.599 | 157 |
| table tennis | 0.535 | 0.589 | 0.561 | 141 |
| tennis | 0.516 | 0.359 | 0.423 | 131 |
| volleyball | 0.750 | 0.449 | 0.562 | 147 |
| accuracy | | | 0.616 | 2742 |
| macro avg | 0.640 | 0.607 | 0.601 | 2742 |
| weighted avg | 0.637 | 0.616 | 0.605 | 2742 |



Figure 7: Analysis and Confusion matrix of SVC_OVR model, test on the left, train on the right

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| baseball | 0.143 | 0.200 | 0.167 | 10 |
| basketball | 0.333 | 0.100 | 0.154 | 10 |
| bmx | 0.250 | 0.100 | 0.143 | 10 |
| football | 0.273 | 0.300 | 0.286 | 10 |
| formula 1 racing | 0.333 | 0.300 | 0.316 | 10 |
| frisbee | 0.222 | 0.200 | 0.211 | 10 |
| golf | 0.400 | 0.200 | 0.267 | 10 |
| hockey | 1.000 | 0.400 | 0.571 | 10 |
| motorcycle racing | 1.000 | 0.300 | 0.462 | 10 |
| rugby | 0.000 | 0.000 | 0.000 | 10 |
| skydiving | 0.102 | 0.500 | 0.169 | 10 |
| snow boarding | 0.000 | 0.000 | 0.000 | 10 |
| sumo wrestling | 0.429 | 0.300 | 0.353 | 10 |
| surfing | 0.636 | 0.700 | 0.667 | 10 |
| swimming | 0.167 | 0.300 | 0.214 | 10 |
| table tennis | 0.500 | 0.100 | 0.167 | 10 |
| tennis | 0.138 | 0.400 | 0.205 | 10 |
| volleyball | 0.000 | 0.000 | 0.000 | 10 |
| accuracy | | | 0.244 | 180 |
| macro avg | 0.329 | 0.244 | 0.242 | 180 |
| weighted avg | 0.329 | 0.244 | 0.242 | 180 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| baseball | 0.231 | 0.425 | 0.300 | 174 |
| basketball | 0.464 | 0.308 | 0.370 | 169 |
| bmx | 0.425 | 0.243 | 0.309 | 140 |
| football | 0.286 | 0.272 | 0.279 | 191 |
| formula 1 racing | 0.524 | 0.398 | 0.452 | 191 |
| frisbee | 0.323 | 0.241 | 0.276 | 133 |
| golf | 0.521 | 0.236 | 0.325 | 157 |
| hockey | 0.889 | 0.186 | 0.308 | 172 |
| motorcycle racing | 0.778 | 0.265 | 0.395 | 132 |
| rugby | 0.786 | 0.066 | 0.122 | 166 |
| skydiving | 0.154 | 0.691 | 0.252 | 149 |
| snow boarding | 0.684 | 0.222 | 0.335 | 117 |
| sumo wrestling | 0.667 | 0.316 | 0.429 | 133 |
| surfing | 0.455 | 0.535 | 0.492 | 142 |
| swimming | 0.407 | 0.503 | 0.450 | 157 |
| table tennis | 0.600 | 0.319 | 0.417 | 141 |
| tennis | 0.149 | 0.473 | 0.227 | 131 |
| volleyball | 0.412 | 0.048 | 0.085 | 147 |
| accuracy | | | 0.319 | 2742 |
| macro avg | 0.486 | 0.319 | 0.323 | 2742 |
| weighted avg | 0.485 | 0.319 | 0.323 | 2742 |

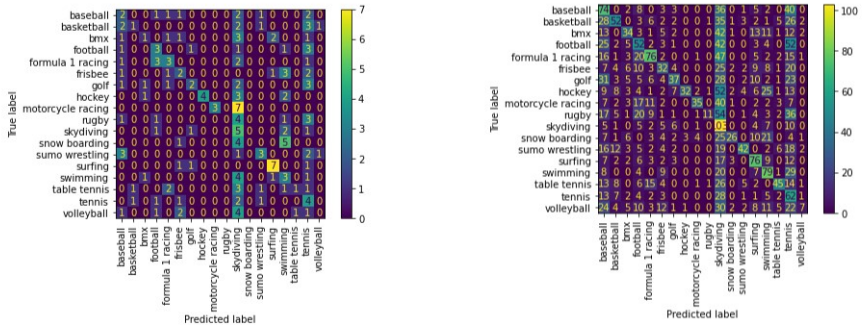



Figure 8: Analysis and Confusion matrix of KNN model, test on the left, train on the right

As an example we can have a look at the confusion matrix of the FT model (fig. 4). We can notice the number 15 on bottom left of the matrix, which implies that the tool will be less useful when predicting volleyball. In fact what that number does is reducing recall on volleyball(last row), predicting some of them as basketball, which as said before would be acceptable if when predicting basketball (false positives) but not when predicting volleyball (false negatives); on the other hand the low numbers on first row show an higher recall on basketball.

It's worth knowing that many parameters combinations have been tested on the SVC_OVO,SVC_OVR and KNN models, but the results look more or less the same. From these results we can see that those methods are performing really bad, and also that they're all overfitting as test accuracy is always less than train accuracy. Among those 3 model both SVC models perform a little better. We can see that the confusion matrices are a little messy but SVC can probably be useful at least in one case. In fact, 'hockey' has 1.0 recall on test set and 0.89 on train set (fig. 6) so it could give some good results.

CNNs instead are working pretty good and as expected FT model performs a little bit better than FE on train set. We can also notice that this gap is a

little bit bigger when we trained on more images, as for larger data fine tuning is less prone to overfit.

## 5.2 Real case scenario

As FT and FE models seems to have good performances I've used those 2 for real cases. I fed as input of the script a 9 minutes video containing just 3 moments where a jump with snowboard was performed. One of the parameter of this script is the threshold of confidence that we want. For FT, in order to have good result, I had to set the threshold to 0.993 which is too much precise for the purpose, while on FE model 0.85 was good enough to have the same result. A threshold below those would result in also having background frames as valid; in both cases we got 4 moments instead of 3. First 3 are jumps and last one is a guy walking. I use the term "moments" to indicate different periods of time as in reality 10 frames per second are extracted, so we have multiple valid frames for each moment.

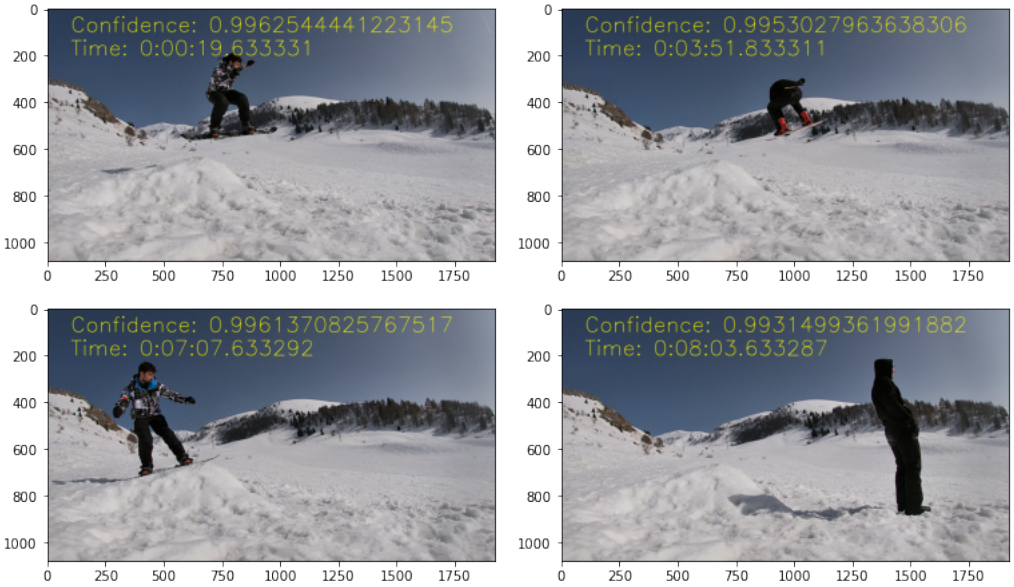



Figure 9: A frame for each moment extracted with FT model

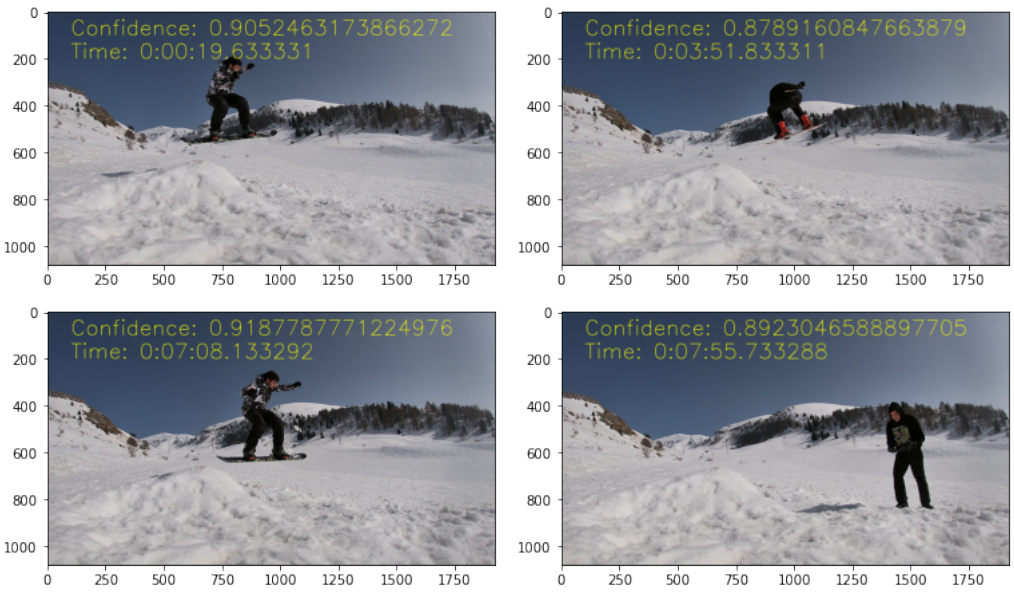



Figure 10: A frame for each moment extracted with FE model

## 5.3 Real case scenario n.2

Another model class is "motorcycle racing" and as I have a personal video related to that I've ran another test. A premise for this case: train images for this class depict road racing motorcycles and just a few dirt bikes, instead the video recorded a "pit bike" which basically looks like a dirt bike but smaller; also the camera was sometimes moved to other places with different angles and light conditions. In order to get some results for this video I had to set the threshold at 0.5.

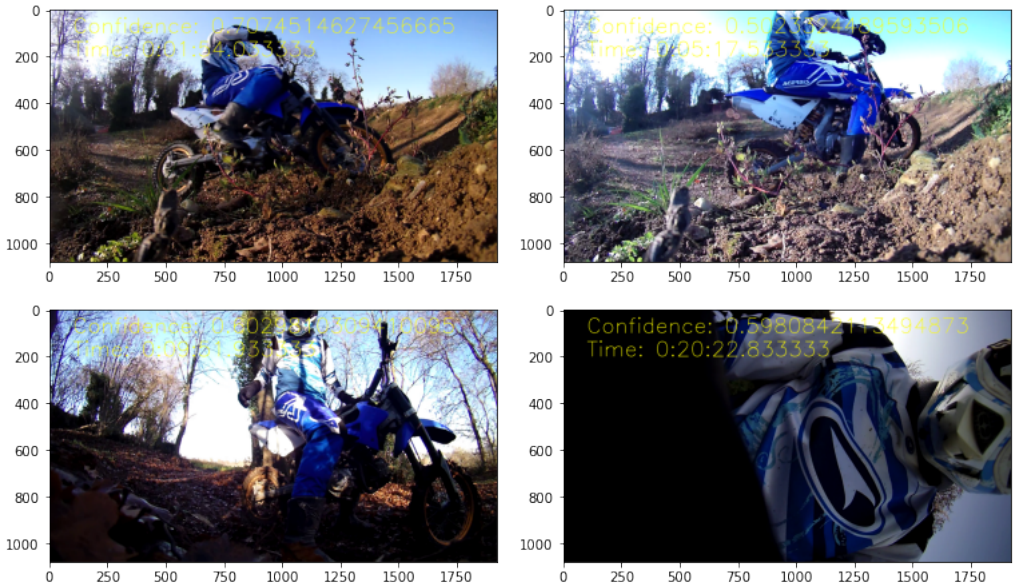



Figure 11: Some pit bike frames

The bike is going around a track so it passes in front of the camera at each lap and just about half of those moments have been recognized. For some minutes the camera was placed in a darker place and there the bike was never spotted by the script. Moreover in many scene the bike is actually visible but far from the camera and so also those moments have not been classified.

# 6 Conclusion

For the initial purpose I had for the project (first real case scenario) I would say that results are not so bad. I got the expected results from my videos and having some false positives is not a huge problem. For the second one instead the results are quite not good: there are no false positive but there are a lot of missing frames and probably in order to achieve some better result I should train again with a dataset of pit bikes not just motorcycles. Another aspect to consider is that images are reduced in dimension before training so if the bike is too far from the camera it won't be recognized. Also the camera is positioned on the ground and angled up and that's a lot different from angles in training set images.

There are two majors disadvantages: the script is limited on classification of only the trained classes, so it doesn't work as an auto cutter for all static video; moreover the script is very slow (around one minute of analysis for each minute of the video) and resource consuming which is not the best for a tool that should analyze hours of videos.

We can think of a more efficient tool that works slightly different: we should start by using some "light" computer vision techniques to remove the static frames of the video and eventually apply the classification on the remaining frames to search for a specific class. Also for the purpose of auto cutting we could skip some frames after one is found as they probably are part of the same action.

# References


[1] *73 Sports Image Classficication*. URL: `https://www.kaggle.com/gpiosenka/sports-classification`. (accessed: 13.09.2021).

[2] *TRANSFER LEARNING FOR COMPUTER VISION TUTORIAL*. URL: `https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html`. (accessed: 07.09.2021).