Classifying Humans Activities Using Human Poses Key-Points

Caroline Dakouré[1], **Camille Felx-Leduc**[1], **Helgi Tomas Gislason**[1], **Nicolas Lemieux**[2], **Jérémy Trudel**[1]
[1]Computer Science, Université de Montréal, Montreal, Quebec
[2]Electrical Engineering, École de Technologie Supérieure, Montreal, Quebec

Abstract

Human activity recognition from multimedia material is a popular problem at this moment. In this paper we will explore how well we can perform on this task using body joints key-points annotation as input for our classifiers with two different data sets and compare these results to a well known CNN architecture namely AlexNet. A third data set, used along with a certain preprocessing technique will then inform us about the importance of temporal information for this task and reveal interesting result about the performance of classifiers.

1 Introduction

Human pose and action estimation has been a core problem of computer vision for the past decade [1]. Intuitively we know that actions are usually characterized by a succession of body positions, but sometimes only a single frame is enough to guess what activity the subject is performing. Yao *et al.* states that using skeleton data alone for action recognition can perform better than using other low-level image data [2]. Based on that, the goal of this project was to characterize, with the help of classifying algorithms, how much the body position of one person could be correlated to the activity they are performing and to investigate how much the spatiotemporal component influenced these results. Spatial points associated with body markers such as an ankle, knee, elbow, head and so on were used to complete this task. The main hypothesis here is that current state of the art (classifying algorithms coupled with the joints annotations) will give us decent results for the classification task. In the scope of this project, three data sets were gathered, which allowed us to investigate the problem from different perspectives. On all three data sets the following algorithms were applied: K-NN, Random Forest and AdaBoost, on top of which some exploration work have been made with SVM and CNN on punctual data sets. For a small description of each algorithm used, please refer to appendix A.

2 Data sets description

2.1 UTD-MHAD

Multimodal Human Action data set [3] contains 27 labeled activities performed by 8 different subjects. Each action is performed 4 times by each subject. Minus three corrupted sequences, this makes a total of 861 data sequences. Each data sequence is represented by a video (.avi) plus matlab data files (.mat) describing the depth, skeleton and inertial sensor data associated with each frame.

2.2 MPII Human Pose

MPII Human Pose data set [4] is composed of 25k images of people doing every day activities. Each image contains one person or more, for a total of around 40k people displayed across the dataset. There are 410 different activities that are split into 40 broader categories. It is possible to learn

either task (categories or activities). Classifying activities is much more difficult due on one side, to the limited number of examples and on the other, to the high number of classes. For each image, there is an associated annotation file that gives the position of the human joints on the corresponding image. These annotations is the result of hand labeled images and was historically created to train and evaluate the performances of automatic segmentation algorithms such as the one we used to annotated our next data set namely Stanford 40 Actions.

2.3 Stanford 40 Actions

Stanford 40 Actions data set [5] contains 9532 images of humans performing 40 different actions. There are between 180 and 300 images per action. Each image has a file associated that gives the coordinates of the bounding box, identifying the humans in each picture. This set does not contains any joints annotations, so we generated them ourselves with the help of OpenPose[2].

3 Preprocessing

3.1 UTD-MHAD

Pre-processing for this database was done by computing the Covariance of 3D joint descriptor, a technique described by Girshick *et al.* [1]. The idea is to compute the covariance matrices of the joints through a sequence. First, we align the 20 joints 3D coordinates from each frame in a 60x1 vector that is denoted by S, then the covariance matrix for that sequence is computed by:

$$C(\mathbf{S}) = \frac{1}{T-1} \sum_{t=1}^{T} (\mathbf{S} - \bar{\mathbf{S}})(\mathbf{S} - \bar{\mathbf{S}})'$$

where $\bar{\mathbf{S}}$ is the sample mean of $\mathbf{S}$, and the ' is the transpose operator. As the covariance matrix is symmetric, we take all the elements of the upper matrix to make the 3D joint descriptor (that is in this case an $60(60+1)/2 = 1830$ elements vector) on which the classification task can be performed. Although as noted by the authors :

"The 3D cov descriptors captures the dependence of locations of different joints on one another between frames during which an action is performed. However, it does not capture the order of motion in time. Therefore, if the frames of a given sequence are randomly shuffled, the co-variance matrix will not change. This is in fact problematic when two activities are in the reverse temporal order of one another".

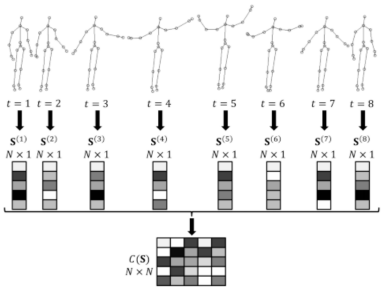

Figure 1: Cov3DJ descriptor visualization as presented by [6]

3.2 MPII Human Pose

Most of the pre-processing was about building a good data structure where each activities were matched with the body joints annotations. Also, different kernels where created such as Minkowski's (p=1,2,3) indicating the distance between each pair of joints, or the cosine metric providing information relative to the angles between each pairs of joints.

3.3 Stanford 40 Actions

We started by cropping the images to the bounding box given along the data set as we wanted to concentrate our identification on the annotation joints. Afterwards, we generated the joints annotations by running a state of the art automatic body annotation tool named OpenPose [7][8][9]. OpenPose renders the segments connecting the joints directly on the image and provide the joints position on the images, that we used for the classification task.

4 Results

Here we will go into details on how well each of the classifiers we implemented performed. Table 1 below presents a summary of the performance and settings of best result for each classifier. For all of our classifiers we used 80% of the data at hand to train and 20% to test. The results of the CNN are based on a training with 20 classes.

Table 1: Overview of the performance of each classifier

Algorithm	Datasets				
	Hyper-parameters	MPII		Stanford	UTD
		Categories	Activities		
KNN	K	1	1	17	1
			5%	25.5%	90.00%
RandomForest	n_estimators	200	200	200	200
	max_depth	15	15	12	7
	max_features	9	9	10	549
	min_samples_leaf	1	1	2	2
		57.96%	49.00%	62.51%	76.65%
Adaboost	n estimators	500	500	5000	150
	learning rate	0.001	0.001	0.001	0.1
		24.80%	8.00%	20.00%	61.85%
CNN- AlexNet*	lr = 0.001		W Annotation	58,59%	
			W/O Annotation	65.14%	
Kernel SVM	γ = 0.0024 c = 42				92.00%

4.1 k-NN

The only hyperparameter in a k-NN classifier is the number of neighbours considered to make a prediction. Hence the strategy to find the best value for K simply was to make a validation set from the training set. The algorithm was then trained on the remaining examples of the training set for a range of possible values of K. In our case we investigated value between 1 and 20. The best value of K was then defined based on the performance on the validation set. The system was then be retrained to include validation examples before assessing the performances on the test set. On MPII the best value for K was 1, on Stanford 40 the best value was 17 and on the UTD-MHAD data set the performances where equal and best for all values of K between 1 and 10, hence the value of K was chosen to be 1 to reduce the computational cost when performing the prediction task.

4.2 Random Forest

To increase the test accuracy for Random Forest, we performed a grid search on all the three data sets, looking for the best hyperparameters. The hyperparameters we investigated were:

- The maximum depth of the Random Forest classifiers (sklearn's max_depth).
- The minimum number of examples at each leaf (sklearn's min_samples_leaf).
- The maximum features each tree used to train on (sklearn's max_features).

We fixed the number of estimators (sklearn's n_estimators) to 200. In the graphs that follow, for Random Forest, we always see the same pattern:

- Test accuracy increases when raising sklearn's max_depth.
- The algorithm reaches a plateau of test accuracy when max_depth gets high.
- We get better test accuracy when raising max_features but not as importantly as max_depth.
- We get better results when keeping min_samples_leaf low.

Another factor of interest is that the Random Forest algorithm always overfitted on our data sets. Training accuracy always reached 100% with best sets of hyperparameters. This can be seen in the *RandomForestHyperParameters*.csv* files in the Results folders with the source code. Those files contain all the combinations of hyperparameters we tried for each data set using Random Forest.

The following graphs introduce our results when tuning max_features, max_depth and min_samples_leaf. For MPII we worked with categories rather than with activities. In fact, both classification problems gave us very similar graphs. We can see that as max_depth increases, so does the test accuracy. Note that these particular graphs don't contain the best set of hyperparameters we achieved for MPII. We saw that if we increased the max_depth further (we went up to 15) we reached an even better test accuracy of 57.97% as mentioned in the first table in section 5.

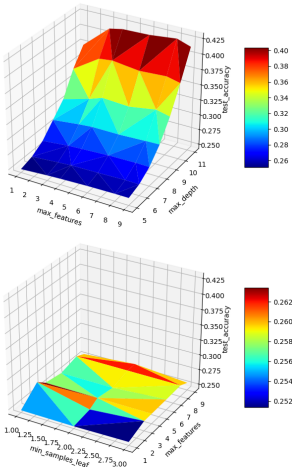

Figure 2: MPII's Random Forest grid search results

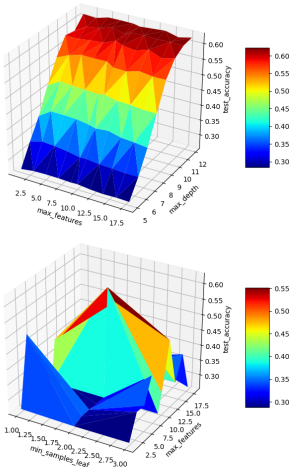

Figure 3: Stanford's Random Forest grid search results

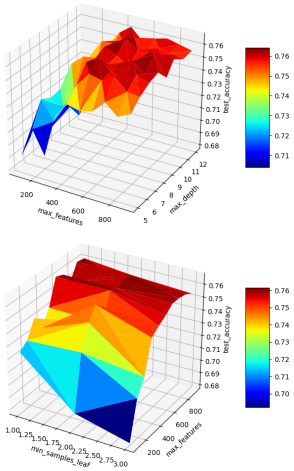

Figure 4: UTD-MHAD's Random Forest grid search results

4.3 AdaBoost

Adaboost was mainly optimized on the MPII data set. The following process will describe the approach we took for the MPII data set only. We based our optimization on two hyperparameters:

- n_estimators: the number of weak learners
- learning_rate: shrinks the contribution of each classifier

We have tuned the hyperparameters with a GridSearch using the following principles:

The number of weak learners should be big enough. The learning rate should be small but the smaller it is, the more we will need weak classifiers. There is a trade-off between the learning rate and the number of weak classifiers.

Finally we chose a grid with a number of estimators between 500 and 2000 and a learning rate between 0.0001 and 0.01. Figure 5, is a graph showing the validation set score (obtained by cross validation with 5 splits) according to the different hyper-parameters.

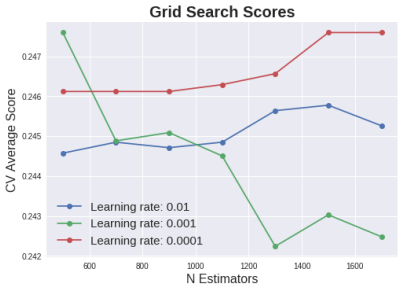

Figure 5: Average validation score based on the hyperparameters for MPII dataset

This graph is interesting because it shows us in more detail the relation between the number of weak classifiers and the learning rate. For the learning rate = 0.001, the curve does not seem to have reached its maximum. We could have done a new gridsearch with a learning rate of 0.001 and a smaller number of weak classifiers to further improve our results.

We had the intuition that distance metrics would give better results on MPII data set than the original coordinates of joints. With this idea in mind, we computed the euclidean distance, minkowski (p=1 and p=3) and cosine.

However, we did not do any new gridsearch,it was only done for the coordinates of the joints and not for the distances. We based the hyperparameters of the distances data set on the best hyperparameters found on the coordinates of the joints.

Surprisingly, the results were close and the coordinates of the joints gave the highest score.

Table 2: Mean score using Cross-validation (CV=5) on MPII with distances

	n_estimators	learning_rate	Mean Score using CV=5
Joints Coordinates	500	0.001	0.248
Euclidean dist.	500	0.001	0.238
Minkowski dist. (p=1)	500	0.001	0.236
Minkowski dist. (p=3)	500	0.001	0.229
Cosine dist.	1000	0.001	0.244

4.4 SVM

The SVM classifier on the UTD-MHAD with temporal relevant data gave the best results for the action recognition classification task. As we can see on the confusion matrix below most of the confusion are on adjacent activities which are more likely to be similar activities with different frame order such as draw circle clockwise (9) and counter clockwise (10) or sit to stand (24) and stand to sit (25).

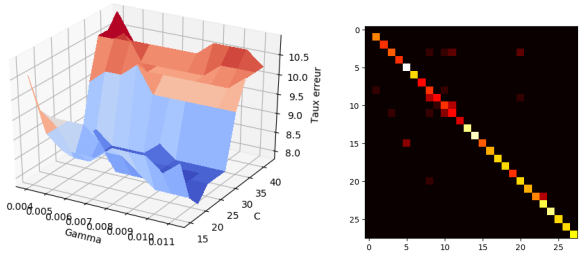

Figure 6: Test set error rate in relations with hyper-parameters and confusion matrix heat-map

4.5 CNN

For learning purposes, we tried a CNN on the Stanford database. Due of some limitations of hardware, we did use a pre-trained model: AlexNet [10]. This network was train on ImageNet database [11], for more than a thousand images. We did this implementation on MatLab [12], because we use transfer learning, we didn't initialize the parameters.

It's important to say that we modified the AlexNet Network, we did change the fully connected layers and the softmax classifier for a 40 classes instead of a 1000 classes.

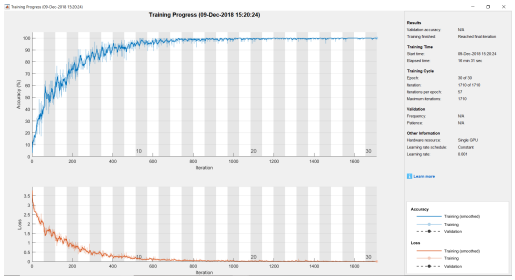

Figure 7: CNN - AlexNet with annontations

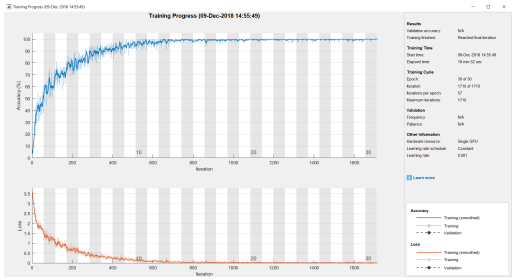

Figure 8: CNN - AlexNet without annontations

First, the stanford database was pre-processed. Then for the purpose of validate the result, we train two different networks. One with the original images (Fig. 8) and one with the pre-process images with OpenPose (Fig. 7). During the phase of training, we changed the hyper parameters, finaly the best results was with a learning rate of 0.001, an epochs of size 30 and an minibatchsize of 64.We did training on 20 classes only, and they were split in 80/20 for the training and the test.

For the images that were pre-process with OpenPose, we trained over the images with the skeleton on the images. We got a result of 58.59%. The graphs of the accuracy and the loss of the training set

are showed on figure 7. Also, we trained over the original images. We got a result of 65,14%. The graphs of the accuracy and the loss of the training set are below (figure 8).

5 Discussion

Preprocessing is often a key part of any classification task. In this project, the challenges where big at this level. As a matter of fact, using a data set in a different direction than for the purpose for which it was created (MPII) composed certain challenges, also being able to implement automatic segmentation with OpenPose on the Stanford data set and achieving similar results to hand labeled ones (MPII) is clearly encouraging and a genuine original contribution that we made in this project. Although multiple strategies were implemented to take advantage of the data, the time constrain also forced us to do some compromises and leave some hypothesis unexplored. Elements that could have been improved regarding preprocessing is to find a better way to normalize the inputs, for example we could have expressed body poses with respect to a certain metric such as the head to hips distance, hence reducing the importance of the variation of the different body shapes in people. Although random forest seemed to perform well even with the raw data.

In all of our grid searches, for that algorithm, we saw that it was overfitting dramatically on the training data. For the optimal combinations of hyper-parameters for each data set, we got 100% training accuracy. At the same time, the testing accuracy would be 60% (Stanford) or 80% (UTD-MHAD). This could have been mitigated by pruning the RandomForest after training until the training accuracy reaches the test accuracy. However, the library we used, sklearn, does not support pruning.

For Stanford 40 Actions, the best classification performances where achieved using a CNN on unlabeled images with 65.14 % although similar results were achieved using only joints key-points and random forest classifying algorithm (62.51%) both greatly outperforming the other classifiers. This demonstrates that pose key-points are in fact a good representation technique. Surprisingly, the euclidean distance and absolute coordinates perform very similarly with only a marginal difference between them. This goes against our initial intuition that that the joints annotations would struggle with prediction accuracy as they are vulnerable to translation and rotation.

Another important result is the performance achieved by the SVM (92%) on the UTD-MHAD data set, showing clearly the importance of pre-processing method and temporal information in action classification task along with the strength of this basic classifier. It out performed more complex models in context such as when data is almost linearly separable and inputs vectors are high dimensional ones. The lack of training examples could also be one of the reason why Random Forest didn't performed as well or better on the UTD-MHAD data set.

6 Conclusion

The results clearly show that even-tough it is possible to proceed on the classification task on still images, adding a temporal component to the analysis/pre-processing definitely increase the success rate on the classification task. An idea for further research would be to incorporate it by creating a grammar of visual positions via clustering strategies and treating the problem with syntactical algorithms.

Finally two results are striking in the work, one is the ability of the Random Forest algorithm to generalize and make sens of the data from only the still images with annotated body poses, performing almost as well as a CNN, well known and widely spread for image recognition for some time now. The second being that with the Cov3DJ processing used along simple classifiers achieved very good performances, mostly being mistaken on time invariant activities such as drawing a circle clock-wise and counter-clock wise.

7 Acknowledgements

Nicolas Lemieux pre-processed all the three datasets, ran the kNN algorithm on two datasets (UTD and MPII) and the SVM on UTD. Helgi Tomas Gislason ran and reported on the grid search for the Random Forest algorithm on all three datasets along with its confusion matrices. Jérémy Trudel

worked on Random Forest and report formatting, Caroline Dakouré ran AdaBoost on all datasets and reported on a grid search for MPII, Camille Felx-Leduc ran the kNN and CNN on Stanford.

References

[1] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, "Efficient regression of general-activity human poses from depth images," in *2011 International Conference on Computer Vision*, pp. 415–422, Nov 2011.

[2] "Does human action recognition benefit from pose estimation?,"

[3] C. Chen, "Utd-mhad: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor," 09 2015.

[4] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele

[5] B. Yao, X. Jiang, A. Khosla, A. L. Lin, L. Guibas, and L. Fei-Fei, "Human action recognition by learning bases of action attributes and parts," in *2011 International Conference on Computer Vision*, pp. 1331–1338, Nov 2011.

[6] M. Hussein, M. Torki, M. Gowayyed, and M. El Saban, "Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations," 08 2013.

[7] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *CVPR*, 2017.

[8] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *CVPR*, 2017.

[9] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *CVPR*, 2016.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[12] MATLAB R2018b, The Mathworks, Inc., "alexnet." `https://www.mathworks.com/help/deeplearning/ref/alexnet.html`. Accessed: 2018-12-08.

8 Appendix

A Algorithms

k-NN: Simple classification method that memorizes the training data and gives a prediction for a new data point by finding the k-nearest neighbors around the data point and taking majority vote.

Random Forest: Constructs many decision trees during training and outputs the class that is the mode of the classes. For each tree that is constructed, the algorithm creates a different subset of the feature set used for the tree. When predicting a new example, the algorithm takes the majority vote from all the individual classification trees.

Adaboost: Combines a set of weak classifiers into a strong one. Each weak classifier is trained with the goal of correcting the mistakes made by the previous ones. The training is focused on some of the examples.

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \qquad \text{where } \alpha_t > 0 \text{ and } h_t \text{ is a predictor}$$

By default, the weak classifiers used with AdaBoost (scikit learn library) are decision trees of depth one.

CNN - AlexNet: Basically, AlexNet is a network of eight layers deep, it begin with five layers of convolution and three layers of fully connected. Also, we modified the network to finish with a fully connected layers with 40 classes and also softmax classifier of 40 classes.

SVM with Gaussian Kernel: Finally as insight by the paper on the Cov3DJ method [6] we gave a try to SVM method. For the hyper-parameters tuning effort, the focus was put on both cost and γ .

B Confusion matrices

We looked at which activities our algorithms were confusing for others. Here are the confusion matrices as heatmaps for the RandomForest classifier.

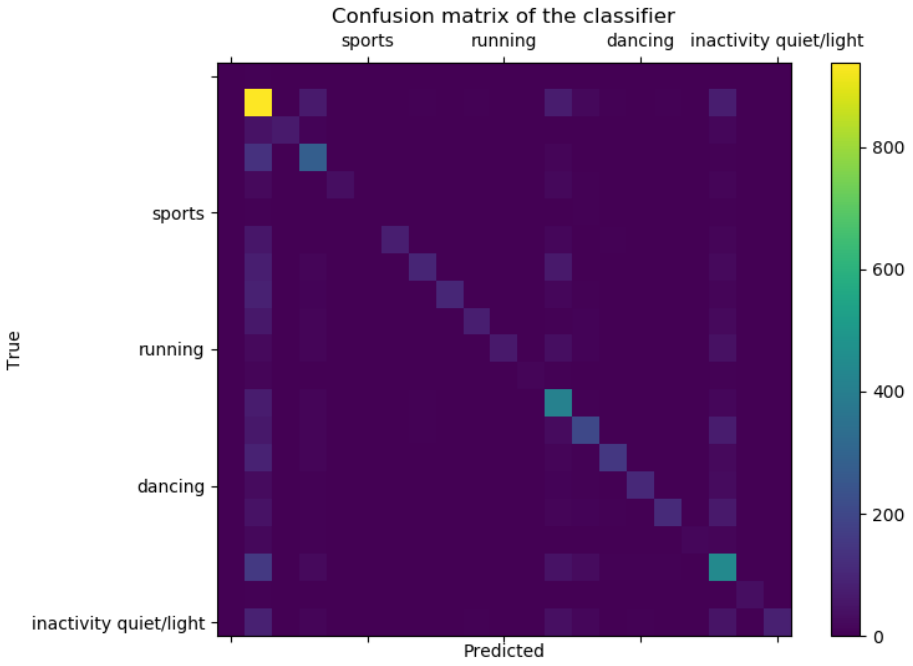

Figure 9: Confusion matrix for the MPII data set. X-axis is the prediction and Y-axis the truth. Interestingly, we see vertical lines of errors. This indicates that every single example could arbitrarily be classified as the same 3-4 wrong classes (the number of vertical lines indicates those classes). There was also one case which was always classified correctly, as shown in yellow.

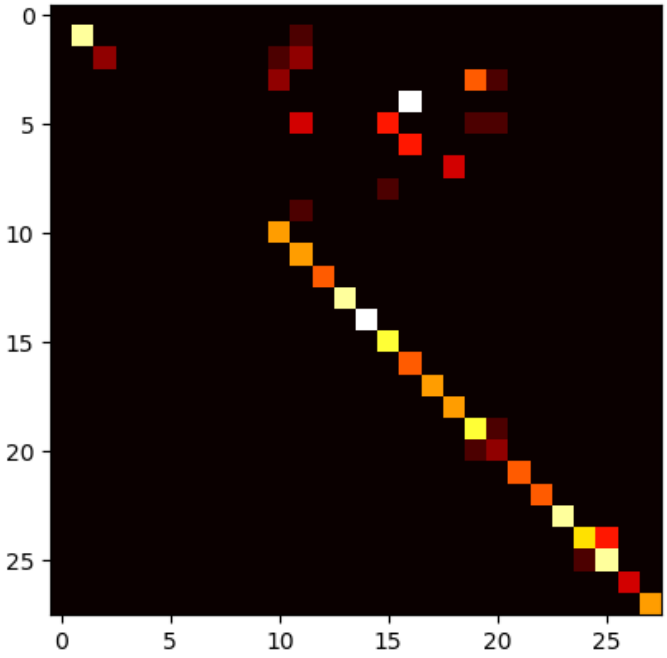

Figure 10: Confusion matrix for the UTD-MHAD data set. X-axis is the prediction and Y-axis the truth. We can see that the RandomForest algorithm usually wrongly thinks that activities 2-9 are activities 10-20. However, the confusion is not symmetrial as activities 10-20 are almost always correctly categorised as 10-20. Activities 2-20 are numerous hand movements (clapping, waving etc.) Seems like the classifier didn't have enough examples of activities 2-9 to make a good prediction on those.

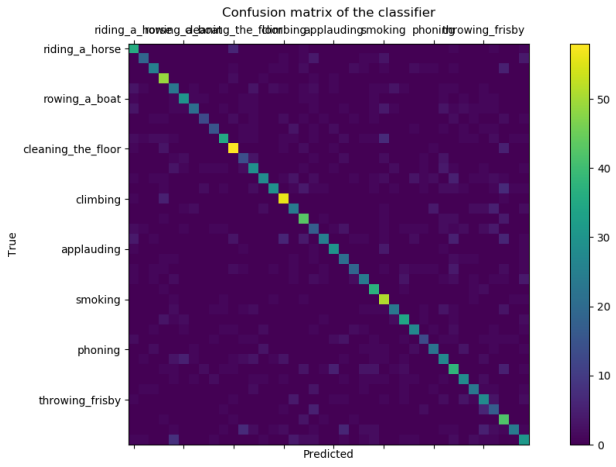

Figure 11: Confusion matrix for the Stanford data set. X-axis is the prediction and Y-axis the truth. Here we see that since we have much more examples, the confusion matrix is a diagonal line. However, since we didn't achieve more than 60% accuracy, the noise is numerous and spread across the matrix.

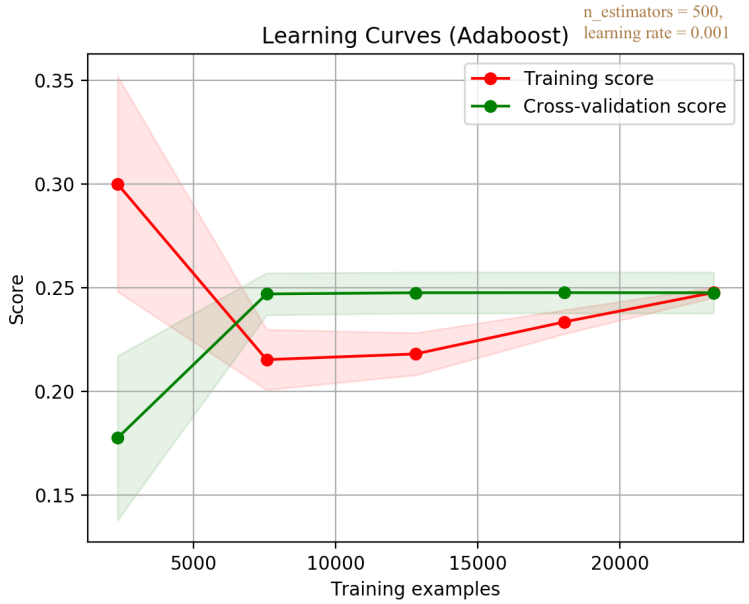

Figure 12: Optimize the number of examples to use for Adaboost and MPII

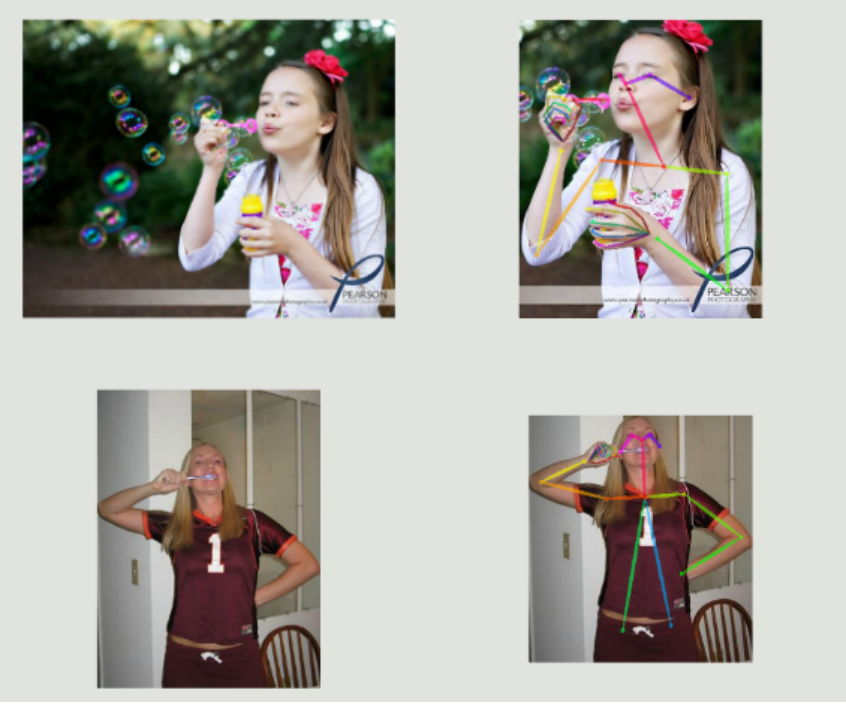

Figure 13: Pre-process results with OpenPose automatic segmentation (croping + joints annotation)