Georgian Artificial Intelligence Networking and Twinning Initiative

EU HORIZON EUROPE - WIDERA-2021- ACCESS-03 (Twinning)
Project #101078950
01.10.2022 - 30.09.25

Automatic recognition of human psychological state on the basis of EEG-based data

Kachiashvili K. Kachiashvili I. and Kvaratskhelia V.

The Sixth IEEE international conference on Image Processing Applications and Systems, 9-11 January 2025 , Lyon, France

Let us consider now automatic emotion recognition on the basis of electroencephalographic (EEG) data set. For the solution of this problem there exist a set of the works where machine learning approach are used, such as:

- Linear Discriminant Analysis (LDA);
- Bayesian Linear Discriminant Analysis (BLDA);
- Support Vector Machine (SVM);
- Multilayer Perceptron (MP);
- Convolutional Neural Network (CNN);
- Hidden Markov Model, k-Nearest Neighbors (KNN);
- Naïve Bayesian (NB);
- Support vector machine (SVM);
- A deep belief network (DBN) and many others.

The best results achieved by these methods are the following accuracies of classification 87.62%, 86.91%, 85.67%, 84.08%, and 69.66% when the classification in two emotional categories (positive and negative) of EEG data from 62-channel electrodes cap were considered.

Despite the rather high accuracy of the obtained results, the mentioned methods of machine learning require *a lot of time and computing resources, which is unacceptable in many practical cases*.

Therefore, the development of algorithms for the solution of the mentioned problem, which allow it to be solved in real time using modern personal computers, remains relevant today and is a very necessary and demanding problem.

The results obtained by us in this direction are presented in the present work.

We consider binary classification of emotion on the basis of 32 dimensional EEG signals. The electroencephalographic (EEG) data were downloaded from the website http://www.eecs.qmul.ac.uk/ mmv/datasets/deap/download.html. For classification into two emotional categories ("high" and "low") of EEG data obtained as multivariate time series (MTS), we used nine well known criteria of similarity of MTS such as (Shahabi and Yan, 2003; Yang and Shahabi, 2004; Mathew, 2023; Lei and Govindaraju, 2004; Gorecki and Luczak, 2015):

- Weighted Sum Singular Value Decomposition (WSSVD);
- Ascending Eigenvalue-Weighted Difference between Eigenvector Matrices (AEWDEM);
- Not Weighed Divergence Between Eigenvector Matrixes (NWDEM);
- Not Weighed Divergence Between Covariance Matrixes (NWDCM);
- Generalized Variance (GV), Descending Eigenvalue-Weighted Difference between Eigenvector Matrices (DEWDEM);
- Principal components analysis similarity factor (PCASF);
- *Eros* (*E*xtended F*ro*benius *s* norm);
- Dynamic Time Warping (DTW).

In addition, two criteria such as:

❖ Getting into the Confidence Regions of the Linear Trends of MTS (GCRLT) and

❖ Union of two AEWDEM (Ascending Eigenvalue-Weighted Difference between Eigenvector Matrices) and Getting into the confidence regions of the linear trends (GCRLT) methods

are introduced and considered in this work.

Formulas for realizing these criteria and a brief explanation of their essence are given in Table.

Table Criteria used for classification of MTS files

N	Criteria	Formulae	Explanation
1.	Weighted Sum Singular Value Decomposition (WSSVD)	$Q_1 = A_1^T \times A_1$, $Q_2 = A_2^T \times A_2$ SVD decompositions: $Q_1 = V_1 \times A_1 \times V_1^T$, $Q_2 = V_2 \times A_2 \times V_2^T$ Columns representation of V_1 and V_2 are $V_1 = [e_1, e_2, ..., e_m]$ and $V_2 = [f_1, f_2, ..., f_m]$, where $A_1 = disg[c_1, c_2, ..., c_m]$, $A_2 = disg[d_1, d_2, ..., d_m]$. The similarity of A_1 and A_2 is defined as: $\theta(A_1, A_2) = \min(\theta_1(A_1, A_2), \theta_2(A_1, A_2))$, where $\theta_1(A_1, A_2) = \left(\sum_{i=1}^{m} c_i (e_i \bullet f_i)\right) / \left(\sum_{i=1}^{m} \lvert c_i \rvert\right)$, $\theta_2(A_1, A_2) = \left(\sum_{i=1}^{m} d_i (e_i \bullet f_i)\right) / \left(\sum_{i=1}^{m} \lvert d_i \rvert\right)$ and $x \bullet y$ is the inner product of the two vectors x and y.	A_1 and A_2 are covariance matrices of MTS to be distinguished;

2.	Ascending Eigenvalue-Weighted Difference between Eigenvector Matrices (AEWDEM)	$WB = \sum_{i=1}^{m} \frac{b(i,i)}{b_1} \sum_{j=1}^{m} (W_1(i,j) - W_2(i,j))^2$, $b_1 = \sum_{i=1}^{m} b(i,i)$	Used MATLAB codes $[a,b,c] = eig(\text{cov\_}matrix)$ and $[V,D,W] = svd(\text{cov\_}matrix)$, where cov_*matrix* is covariance matrix of MTS under investigation; c is the right eigenvector matrix; a is the left eigenvector matrix (the number of columns is the rank of the covariance matrix); b is a diagonal $m \times m$ matrix of the eigenvalues λ_i, where $\lambda_1 \geq ... \geq \lambda_r \geq 0$. The eigenvalues and the corresponding eigenvectors are sorted in non-increasing order.
3.	Not Weighed Divergence Between Eigenvector Matrixes (NWDEM)	$WE = \sum_{i=1}^{m} \sum_{j=1}^{m} (W_1(i,j) - W_2(i,j))^2$	Used MATLAB code $[V,D,W] = svd(\text{cov\_}matrix)$.

4.	Not Weighed Divergence Between Covariance Matrixes (NWDCM)	$WC = \sum_{i=1}^{m} \sum_{j=1}^{m} (A_1(i,j) - A_2(i,j))^2$	$A1$ and $A2$ are covariance matrices of MTS to be distinguished.
5.	Generalized Variance (GV)	$WM = \frac{\left\|A_1^{1/2}\right\| \cdot \left\|A_2^{1/2}\right\|}{\left\|(A_1^{1/2} + A_1^{1/2})/2\right\|}$	A_1 and A_2 are covariance matrices of MTS to be distinguished. This criterion is developed for comparison on the similarity of two normal population covariance matrices.
6.	Descending Eigenvalue-Weighted Difference between Eigenvector Matrices (DEWDEM)	$WD = \sum_{i=1}^{m} \frac{D(i,i)}{D_1} \sum_{j=1}^{m} (W_1(i,j) - W_2(i,j))^2$, $D_1 = \sum_{i=1}^{m} D(i,i)$	Used MATLAB code $[V, D, W] = svd(\text{cov\_}matrix)$, were $\text{cov\_}matrix$ is covariance matrix of MTS under investigation; matrix V's columns are the corresponding right eigenvectors, so that $A*V = V*D$; D is diagonal matrix of eigenvalues; full matrix W's columns are the corresponding left eigenvectors, so that $W'*A = D*W'$.

7.	Principal components analysis similarity factor (PCASF)	$S_{PCA}(A_1, A_2) = trace(PC^T CP^T) = \sum_{i=1}^{k}\sum_{j=1}^{k} \cos^2 \theta_{ij}$, where P and C are the matrices that contain the first k principal components of A_1 and A_2, respectively, θ_{ij} is the angle between the i th principal component of A_1 and the j th principal component of A_2.	A_1 and A_2 are covariance matrices of MTS to be distinguished.
8.	*Eros* (*E*xtended F*ro*beniu*s* norm)	$Eros(A, B, \omega) = \sum_{i=1}^{n} \omega_i \lvert < a_i, b_i > \rvert = \sum_{i=1}^{n} \omega_i \lvert \cos\theta_i \rvert$, where $< a_i, b_i >$ is the inner product of a_i and b_i, ω is a weight vector which is based on the eigenvalues of the MTS dataset, $\sum_{i=1}^{m} \omega_i = \sum_{i=1}^{m} D(i,i)/D_1 = 1$ ($D_1 = \sum_{i=1}^{m} D(i,i)$) and θ_i is the angle between a_i and b_i. The range of *Eros* is between 0 and 1, with 1 being the most similar.	A and B are covariance matrices of MTS to be distinguished. V_A and V_B are two right eigenvector matrices by applying SVD to the covariance matrices, A and B, respectively. $V_A = [a_1, ..., a_m]$ and $V_B = [b_1, ..., b_m]$, where a_i and b_i are column orthonormal vectors of size m.
9.	Dynamic Time Warping (DTW)	$dist = dtw(X, Y)$ stretches two matrices $X_{m \times n_1}$ and $Y_{m \times n_2}$ by repeating their columns onto a common set of instants such that distance, the sum of the Euclidean distances between corresponding columns, is smallest. In that case, X and Y must have the same number of rows.	Used MATLAB codes $dist = dtw(X, Y)$, $X_{m \times n_1} = A^T_{n_1 \times m}$, $Y_{m \times n_2} = B^T_{n_2 \times m}$, where A and B are two matrix of MTS.

10.	Getting into the confidence regions of the linear trends of MTS (GCRLT)	**Input**: C, $S1$, $S2$ - MTSs corresponding to a controlled, standard 1 and standard 2 states, respectively.
		
		for $i=1$ *to* 32 *do*
		
		Isolate: trC_i (trends of controlled MTS)
		$trS1_i$ (trends of MTS, corresponding to the "high" condition standard)
		$trS2_i$ (trends of MTS, corresponding to the "low" condition standard)
		Compute: $\Delta trS1_i$ (deviations from the trends of the "high" condition standard with the given confidence probability)
		$\Delta trS2_i$ (deviations from the trends of the "low" condition standard with the given confidence probability)
		end for i
		
		$I1=0;$
		$I2=0;$
		
		for $i=1$ *to* 32 *do*
		
		if $trC_i \in (trS1_i - \Delta trS1_i; trS1_i + \Delta trS1_i)$
		
		$I1=I1+1;$
		end if
			Attribution of the trends of the control MTS is controlled by the facts of falling within the confidence regions of the trends of the corresponding time series of the two MTSs fixed as standards, and a decision is made in favor of the standard, i.e., of the corresponding state, for which this condition is more satisfied.

		if $trC_i \in (trS2_i - \Delta trS2_i; trS2_i + \Delta trS2_i)$
		$I2 = I2 + 1;$
		end if
		end for i
		if $I1 > I2$
		Print "*high*" *state takes place*"
		else
		Print "*low*" *state takes place*"
		end if
		In more details see Algorithm 3 of Appendix.	

11	Union of two methods AEWDEM and GCRLT		For the control MTS, the proximity to the two standard states ("high" and "low") of the corresponding MTSs is tested simultaneously by the second and tenth criteria, and a decision is made in favor of the state for which the total criterion is more satisfied.

The classification of EEG signals was carried out in different scenarios:
1) MTSs selected to represent "high" and "low" states, i.e. the so-called standards and classifiable MTS belong to the same person;
2) MTSs selected to represent "high" and "low" states, that is, the so-called standard MTSs belong to one group of persons, and classifiable MTSs belong to another group of persons.

The 11th method gave on average the best results for both scenarios. In particular, it gave 81.25% accuracy for the first scenario and 83.33% accuracy for the second scenario. Though, it must be said that the second method gave accuracy 93.75 in one case for the first scenario.

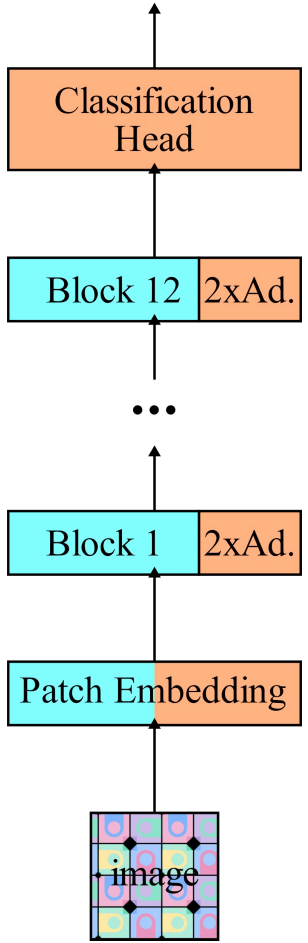
At the same time, it should be emphasized that the implementation of these methods do not require large computing resources and calculation time. It is implemented on a personal computer of average power and a decision is made in a few seconds.

Fine tuning embedding layer along with adapters in vit

Saghinadze T.

Model Description

- While adapters and most other parameter-efficient fine-tuning (PEFT) methods are mostly used for language models, they have demonstrated remarkable success in the vision domain.
- A common practice is to incorporate adapters solely within transformer blocks.
- However, the most frequently cited origin of adapters is Rebuffi adapters, originally designed for Convolutional Neural Networks (CNNs).
- Given that ViT and similar architectures often employ convolutional layers for embedding implementation, it is natural to inquire whether fine-tuning the embedding layer itself could yield performance improvements.
- For our investigation, we utilize ViT-B/16 as the base model, incorporating Houslby adapters alongside several methods for fine-tuning.



Results

DB : cifar-100; Backbone :

CNN Adapter	lr = 0.03	lr = 0.01	lr = 0.003	lr = 0.001
No CNN Adapter	**92.63**	**92.26**	91.69	90.43
Full Embedding	**92.27**	**91.99**	91.5	90.37
CPD-LoRA rank=2	**92.52**	**92.13**	91.64	88.94
TKD-LoRA rank=2	**92.66**	**92.11**	91.71	90.43
Residual Adapter	91.78	91.22	90.3	88.94
Grouped Adapter	91.3	86.65	55.18	26.84
CHAPTER Adapter	83.77	72.99	46.9	26.12

maximum validation set accuracy over 15 epochs at different learning rates

Best result obtained using IMAGENET-1K (300 epochs) 87.8%,
IMAGENET-21k (30epochs) 91.6%, JFT-300M 91.87%

CNN Adapter	r = 2	r = 4	r = 8	r = 16	r = 32	r = 64	r = 128
CPD-LoRA	**92.52**	**92.45**	**92.68**	**92.34**	**92.49**	**92.31**	**92.36**
TKD-LoRA	**92.66**	**92.56**	**92.49**	35.51	32.56	32.25	28.1

maximum validation set accuracy over 15 epochs at different ranks with learning rate 0.03

Results

DB : RESISC-45; Backbone :

CNN Adapter	lr = 0.03 p = 0.2	lr = 0.03 p = 0.0	lr = 0.01 p = 0.2	lr = 0.01 p = 0.0	lr = 0.003 p = 0.2	lr = 0.003 p = 0.0
No CNN Adapter	94.68	94.27	93.10	93.19	86.94	86.70
Full Embedding	94.02	94.17	93.76	92.03	87.24	87.37
Residual Adapter	92.76	91.30	90.75	90.16	85.89	86.08
Grouped Adapter	92.54	90.14	87.98	85.06	72.27	71.73
CHAPTER Adapter	93.06	91.59	89.16	86.78	73.70	72.43
CPD-LoRA rank=4	94.41	94.51	93.08	93.06	86.81	86.43
CPD-LoRA rank=16	94.49	94.33	93.11	93.05	86.57	85.52
CPD-LoRA rank=64	94.60	94.35	91.92	91.21	82.02	84.17
CPD-LoRA rank=256	93.76	94.30	87.35	87.41	-	-

maximum validation set accuracy over 30 epochs at different learning and dropout rates

Best result obtained using IMAGENET-21k (30epochs) 96.1%,

conclusions

- Models with weaker backbone adapters generally give noticeable gains in top-1 accuracy for CIFAR-100.
- Our experimental setup failed to obtain higher accuracy for a more robust backbone for RESISC-45, but this might be due to the absence of a learning rate scheduler.
- Nevertheless, the main objective was to check whether fine-tuning the embedding layer would increase performance.
- In the former case, performance was on par with keeping the embedding layer untouched, and in the latter case, it reduced accuracy.
- These results suggest that fine-tuning solely the transformer blocks/heads may be more effective.

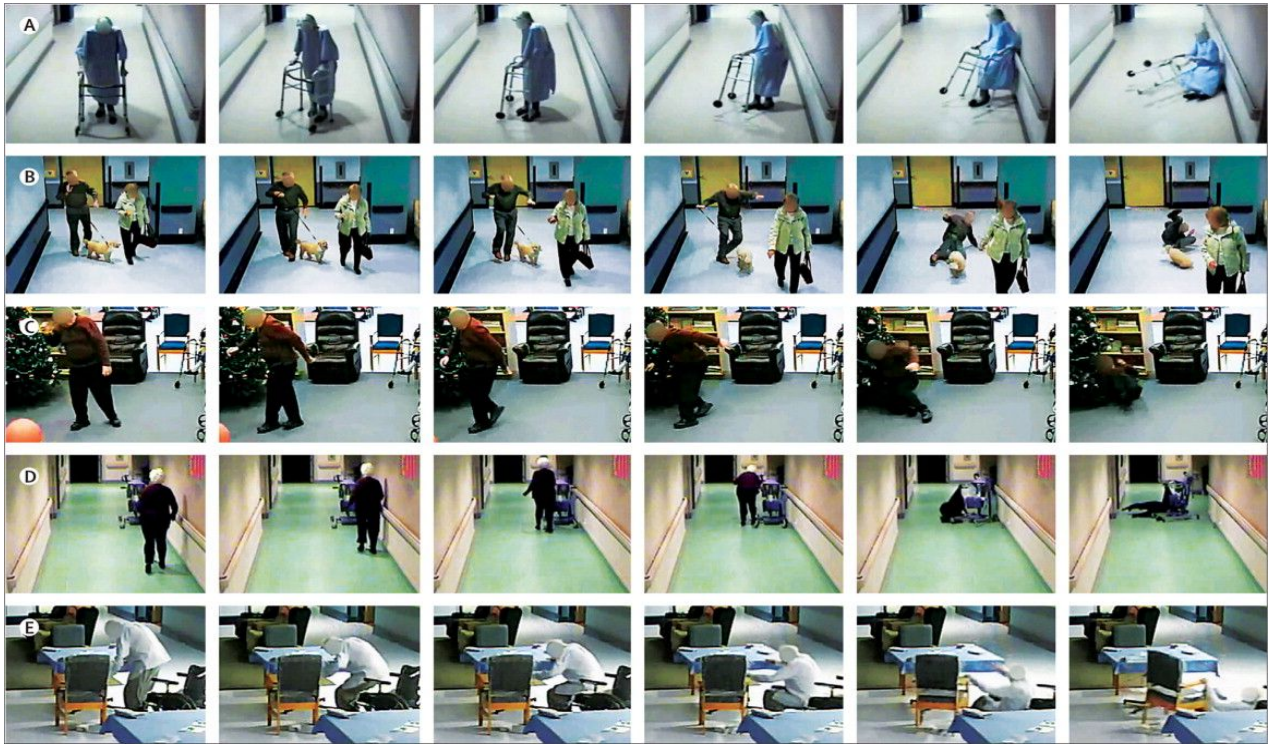
Emotion Recognition from Human Body Movements

Ioseb Kachiashvili, Luka Tabagari, David Karapetian, Philipp Müller, Benedikt Emanuel Wirth, Michal Balazia

Body gestures	Jump	Squat	Throw	Stand	Recede	Turn and walk away
Neutral						
Emotional	Happiness	Sadness	Anger	Surprise	Fear	Disgust
						

Our Goal/Motivation

Our goal is to create and implement a system using full body emotion recognition from multiple views in everyday life.


A
B
C
D
E


Indoor Camera Test


ring.com

Dataset: MPIIEmo Emotion recognition from embedded bodily expressions and speech during dyadic interactions

Philipp Müller, Sikandar Amin, Prateek Verma, Mykhaylo Andriluka, Andreas Bulling
Proc. International Conference on Affective Computing and Intelligent Interaction (ACII), pp. 663-669, 2015.

View	Sub-scenarios			
1				
2				

View	Sub-scenarios			
3				
4				
5				

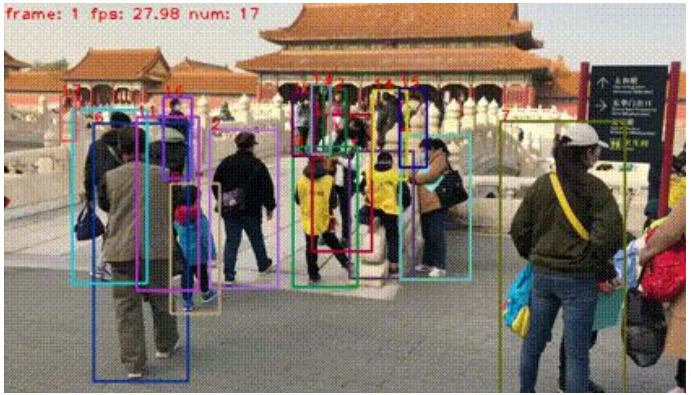
View	Sub-scenarios			
6				
7				
8				

OpenPose

OpenPose is an open-source system for human 2D pose estimation of multiple people.

ByteTrack

- Designed for multi-object tracking (MOT).
- Enhances association accuracy between detected objects across frames.
- Maintains consistent identities over time.

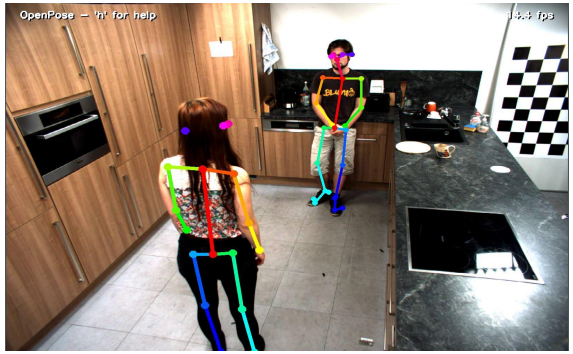


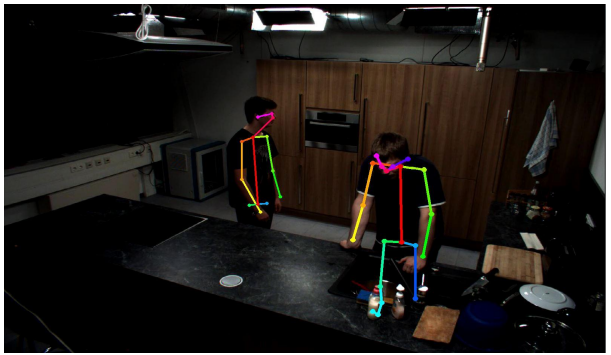
GitHub - ifzhang/ByteTrack: [ECCV 2022] ByteTrack: Multi-Object Tracking by Associating Every Detection Box

Processing the Data

OpenPose

Worked perfectly on the MPIIEmo dataset for detecting key points and drawing the skeleton after.

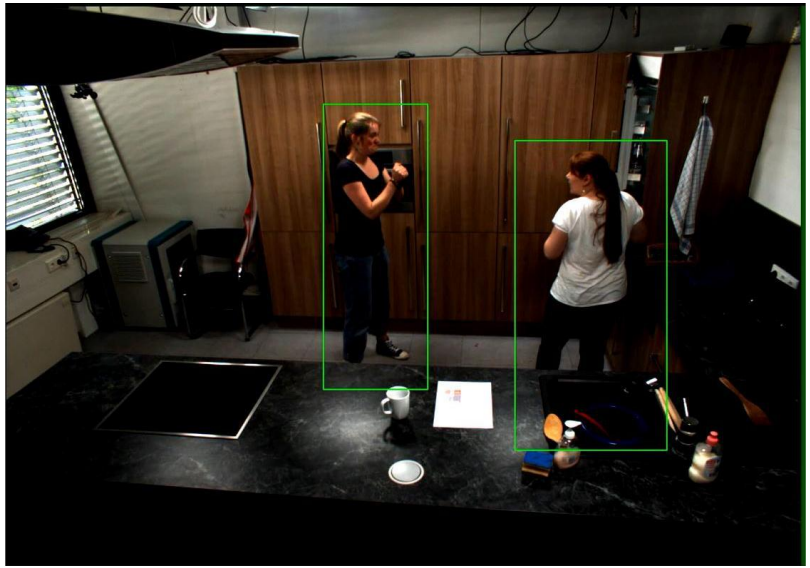




OpenPose

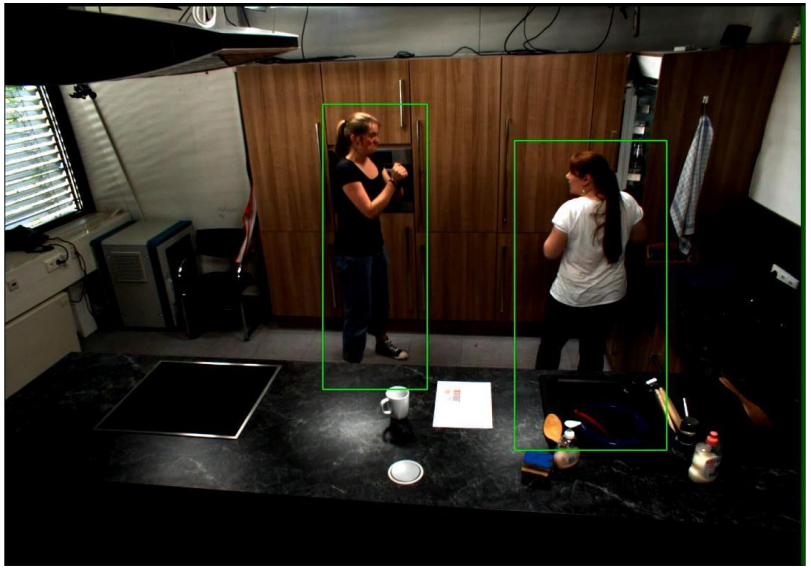
Next steps:

- Draw bounding boxes based on tracking results.
- Crop bounding box image sequences from videos.
- Create separate videos from these sequences.



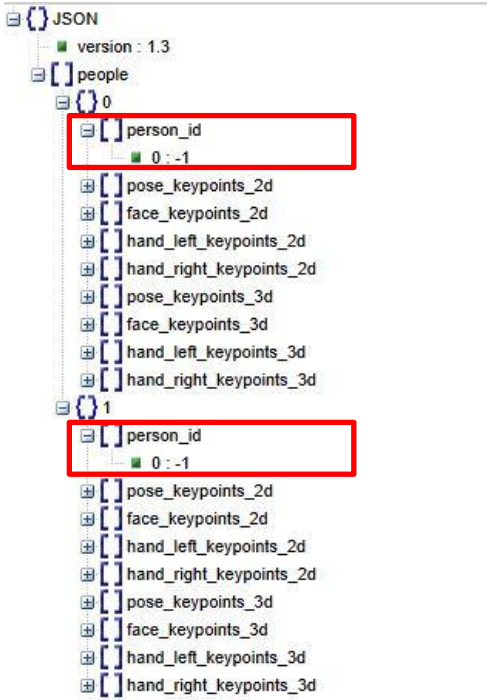
Challenges with OpenPose

OpenPose indeed excels at detecting and tracking human key points, but it does not provide unique IDs for each detected object across frames.



Challenges with OpenPose

This lack of unique IDs means we can't create separate videos from cropped bounding box image sequences.

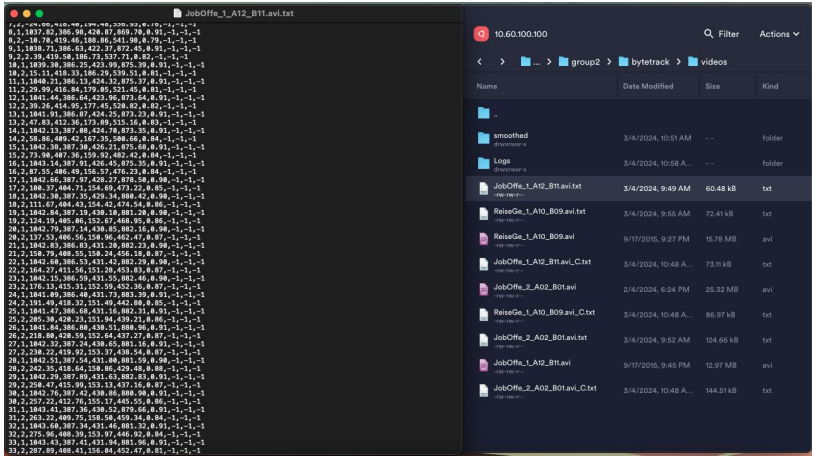


ByteTrack

Process:
ByteTrack generates tracking results and output video files with bounding boxes.

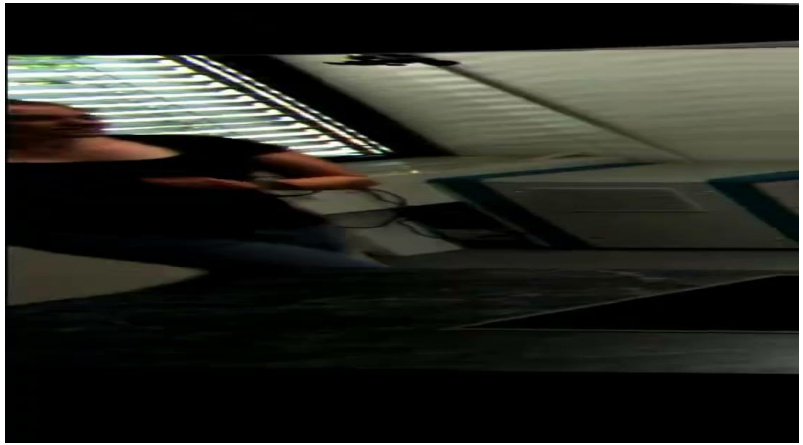
Tracking File Format:

- Frame number
- Unique identifier
- x-coordinate (top-left)
- y-coordinate (top-left)
- Width
- Height



Challenges with ByteTrack

Issue 1:
Cropped separate videos by unique ID's based on tracking results are stretched. Because dimensions for each bounding box in tracking file are different on every frame.



Fix:
Resize all bounding boxes to have the same maximum width and height found in the tracking results, while ensuring that each bounding box remains centered around its original position.

Challenges with ByteTrack

Issue 2:
If an Object is near the camera, ByteTrack would write a very big number for it's dimensions.

After checking tracking results file, we see anomaly in height size:

Frame	ID	X1 AND Y1	width, height	
35,447,803.09,508.11,148.18,458.85,0.92,-1,-1,-1				
35,448,28.76,405.19,516.68,1332.17,0.79,-1,-1,-1				
36,447,802.78,507.90,148.85,460.29,0.92,-1,-1,-1				
36,448,18.56,398.66,533.63,1344.67,0.81,-1,-1,-1				
37,447,802.67,507.80,148.94,459.99,0.92,-1,-1,-1				

Without fixing this, after processing tracking results, all bounding boxes for that id would get same height size.

Challenges with ByteTrack

Issue 2:
If an Object is near the camera, ByteTrack would write a very big number for it's dimensions.

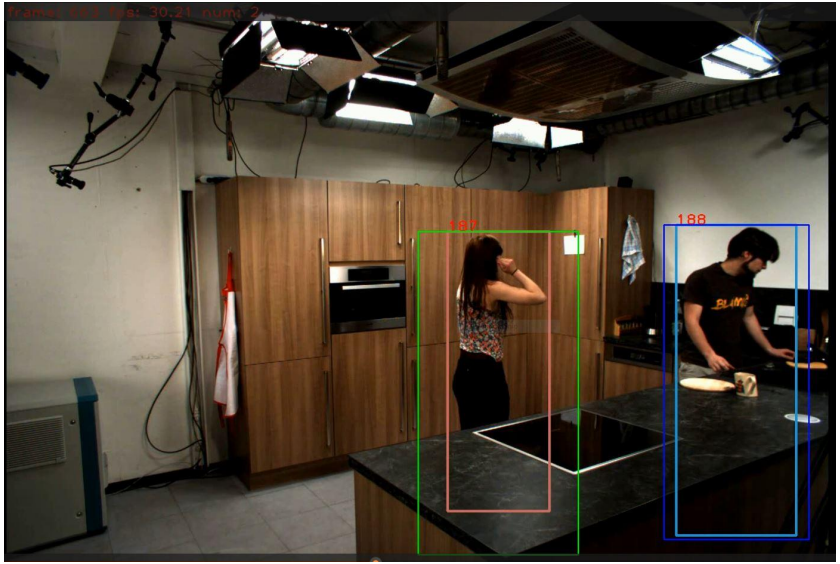
Fix
Solution Steps:

1. Parameter fine-tuning for adjustment process.
2. Modifying height and width to handle extreme outliers.
3. Detecting and adjusting anomalies.
4. Updating bounding boxes while maintaining centre position.

- **Percentile based method to remove possible outliers.**

Challenges with ByteTrack

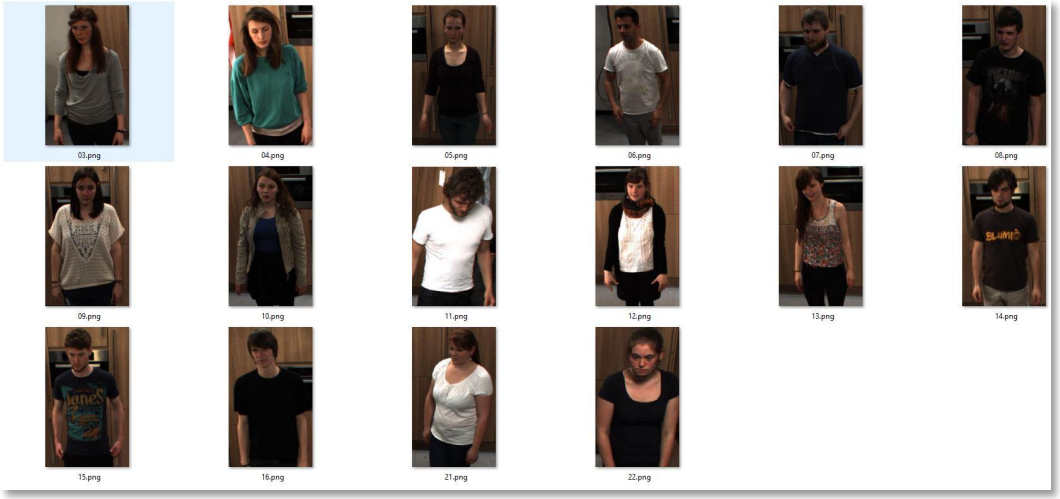
- While experimenting with both, we got our desired results.
- After updating dimensions for bounding boxes, median smoothing was applied and new tracking file was generated .



ByteTrack

Outcomes:

1. Cropped bounding box sequences for each unique ID.
2. Created new video files from image sequences.
3. Annotated videos according to provided instruction file from Dataset.



Thank you!