Pose-based human action recognition from surveillance videos

Advisor: Anh Tu Nguyen
tu.nguyen@nu.edu.kz

Allan Dyussenbayev
allan.dyussenbayev@nu.edu.kz

Alibek Seksenali
alibek.seksenali@nu.edu.kz

Miras Bakytbek
miras.bakytbek@nu.edu.kz

Abstract

Senior project is a web service which represents an action recognition system where users can register, provide video files as an input. All inputs are processed using different deep learning models implemented by our team: Convolutional Neural Network (CNN), DNN, Transformer Neural Net, Long Short-Term Memory (LSTM). Output is a video file, where action labels, human skeleton and frame are drawn for every person present in the video. Also users can specify which kind of actions they are interested in. If those action labels are recognized, then notifications are sent to the telegram chat. Main algorithm uses openpose for feature extraction. Then DeepSort for uniquely identifying people present in the video file. Finally classification is performed with Transofrmer neural net. Also we took LSTM for accuracy comparisons. We took CNN and DNN in Fall but those models were suffering from pose flickering . RNN models, which are LSTM and Transformer handle pose flickering and process sequence of frames rather than in frame by frame fashion. Transformer's accuracy is 96,4 % for the dataset that we created having two people present. LSTM approach demonstrated a 94% accuracy. Transformer is mostly used for Natural Language Processing (NLP), but we demonstrated that it might as well be capable of handling video processing tasks and show high accuracy. Training and testing were done on the dataset that we created by ourselves.

1. Introduction

Modern applications create vast amounts of video and image data. Video/image analysis became a topic of interest of researchers in computer vision. studies were carried out with a focus on action recognition. This problem has many applications in sport analysis, smart surveillance, censorship implementation etc. Let's consider public street surveillance. There might be cases such as crimes involving one group of people harassing the other, people feeling sick falling down and so on. Despite the fact that supervision is provided in many areas in the cities, reliance on humans alone is not effective. People need to receive help as soon as possible, otherwise tragic outcomes might be in place. Action recognition system is able to detect such cases fast and report incidents with little delay. Combination of supervision performed routinely by people with an activity recognition framework makes it more possible for human lives to be saved. Also in sport analysis action recognition systems might be a useful tool to generate data necessary to make conclusions about individual athletes' performance, team characteristics, game outcomes. Activity classification task is complicated due to the presence of factors such as background noises, intra-class variability, frame processing is also costly. Current activity recognition solutions can be categorized into pose based and non pose based. The first type deals with extraction of human features from the input(image/video file). The second analyzes the whole images and video files. It has been shown already that pose based approaches demonstrate higher accuracy but harder to implement due to huge feature preprocessing steps. Current action label classifications are achieved through the use of convolutional neural nets, LSTMs etc. The solution we are offering is a combination of existing solutions for action recognition and tracking. It also has unique components for this problem. Similar to most of the attempts to solve the activity recognition task, we extract human joint positions. That job is carried out by the OpenPose framework. We also made the focus on differentiating people appearing in the image/video by assigning unique ID to every person appearing in the video as shown in figure 1.

This is an object tracking problem. In order to handle that we decided to implement DeepSort [6]. It is smart enough to understand that the person disappearing from the frame and appearing in another one is the same person. DeepSort demonstrates high performance on t–he input having a lot of frames and individuals and low amount of identity switches. Its approach is based on nearest neighbor queries in visual appearance space. Action classification was implemented differently compared to most of

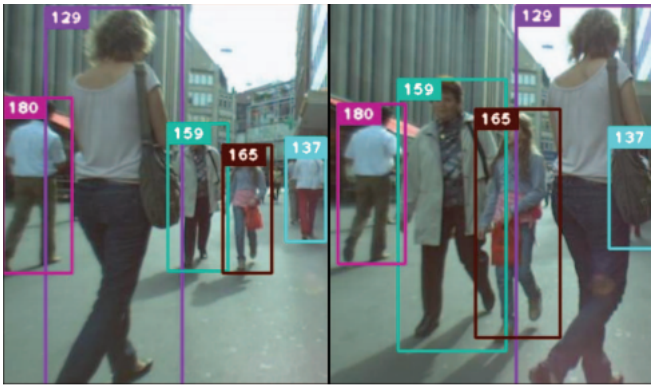

Figure 1. Tracking by DeepSort [6]

the solutions. We adopted the Transformer neural net and compared its recognition accuracy with more baseline approaches which are LSTM and convolutional neural nets. Information above concerns only with a main algorithm but we have more than that. We created a web service with register/login pages. Application allows to easily upload the video files and quickly receive the video output directly in the browser. If action labels specified by the user are identified, the notifications will be sent to telegram.

Report structure is the following:

1. Related work with summary of works related to action recognition task
2. Project narrative where we describe decisions flow regarding main system's components
3. Solution description with a detailed explanation of the way the system's components work
4. Results section explains different metrics for the performance of the classification models along with architecture and results discussion
5. Conclusion is a summary of the whole report.

2. Related work

Our solution is based on the DeepSort algorithm explained in detail in [6]. This object tracking methodology compared to other approaches allows reduce the number of occlusions and identity switches. Research results in the above paper were testes on MOT dataset [4] and showed 45 % reduction in identity switches. Compared to approach used in this project ,[1] used ActionXpose, which is also pose based, however it extracts low and highdimensional features, which are firther provided to LSTM. Application of OpenPose is based on the report [2]. OpenPose was used to extract human joint positions in this work, which were preprocessed afterwards and then fed up to CNN model. Human activity classification task was completed used using Transofrmer neural net in [5]. Compared to our algorithm, their method takes the whole video as input parameter, while we use OpenPose and then deep sort to provide the input to the transformer. This is an encoder based transformer , while we have a decoder based cause OpenPose and DeepSort are responsible for encoding. They tested the algorithm on AVA dataset [3], which consists of spatiotemporally localized atomic visual actions.

3. Project narrative

3.1. Working with features

Starting point of this project was to implement the OpenPose framework for the joint extraction, which were taken as features. Then we preprocessed those features. During Fall semester we took joint positions, discarded frames where the head or thigh are missing, removed joints having head. In the next semester we changed the feature vector. Now we do not take joint positions, but compute angular positions and distances computed from those joint positions. It increased the classification accuracy. Another change is that we refused to remove joints from the head because it turned out to be an important joint.

3.2. Classification

For the Fall semester CNN base-model was implemented , which is non-pose based, and its accuracy results were compared with DNN and RNN models. DNN showed higher accuracy than CNN and suffered from pose flickering due to the imperfections of the DNN model. Pose flickering refers to situations when slight pose change leads to the change of action label. Problem comes from the fact the processing takes place in frame by frame manner.

In the Spring semester DNN was replaced first with LSTM and then with Transformer neural net. Problem described above was significantly reduced due to the fact that those models process sequences of frames. Transformer algorithm has been added to the list, because we expected greater results from it compared to LSTM, because the transformer has an attention mechanism with a higher parallelisation rate and higher performance, and besides all, RNN models take frames sequence,which reduces pose flickering.

3.3. Dataset

Dataset that was created for the Fall semester was made incorrectly. We divided each video into equal segments with 2 seconds duration, while the division had to be handled manually in order to exclude inappropriate inputs. By inappropriate input it is meant cases when , for example, action is started in the beginning of the video and not finished by the end of the video, while another video segment starts

with completion of the action from the past video segment. Also plan made in Fall for the Spring included expanding the list of recognized actions.

CNN and DNN required a small size of processed video files. Due to that brute force solution, some segments included corrupted inputs for training models. We were making it up from scratch several times but it still was far from the sufficient quality. Sufficient quality is when dataset consists of video segments of small duration each having full action performance completion.

The dataset creation process for RNN models had some new specifications and requirements. We created one with one person and another with two people. The main struggle was the requirement of a much bigger size of dataset.Training and testing required a larger one. Problem was solved by combining single and couple datasets. As a result, LSTM and Transformer models had fully met our expectations in terms of accuracy. Our final combined dataset has high quality. Finally, we did not expand the list of recognizable actions, because during the Spring semester we decided that it would widen the scope of our project too much and it would be much harder to focus on the quality. Actually, the list of recognizable actions depends only on the properties of the dataset. It means that to recognize the action the model just needs to have enough training material with performance of this action.

3.4. Web Service

We decided to make a web service for the sake of having a convenient interface to work with our algorithm. It integrates with Telegram bot, which sends notifications to the user in case that some actions are recognized.

4. Solution Description

Figure 2 shows conceptual solution model for the algorithm we are using.

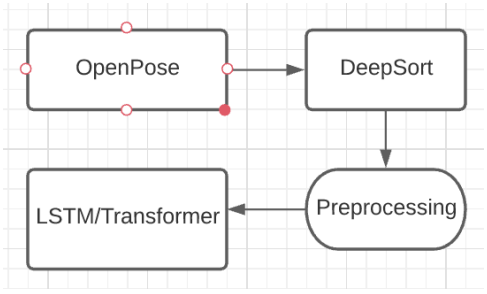

Figure 2. Workflow of the main algorithm components

Workflow description in fig 2 is shown below

1. Joint extraction from the video file with OpenPose with features formation.
2. DeepSort application resulting in feature vector modification with adding ID field , which is assigned to each individual by using obtained features.
3. Feature processing by computing angular joint positions and joint distances.
4. LSTM or Transformer application for action label classification

Next sections will examine how those components work internally and how they interact with one another.

OpenPose

First of all, we used OpenPose as can be seen in figure1. Connected points form the skeleton. This is exactly what OpenPose generates that is everything except for ID, frame,action label.

Figure 3. OpenPose output is connected joints on human body

Input (image/video) is fed up to a two-branch CNN. Then jointly prediction occurs of confidence maps associated with recognition of parts of the body and for affinity fields for association of those parts. Then the parsing step is performed resulting in a set of bipartite matching for human body parts association. Finally, matchings are assembled into complete body poses for all people. After the video/image is provided, the content hoes through the neural net producing a Heatmap and Part Affinity Fields. All HeatMaps are associated with each body part illustrated below.

Part Affinity Fields, on the other hand, provides data regarding the position and orientation of pairs(the connection between parts). Next step is application of Non Maximum Suppression algorithm. Its aim is to detect the parts locations out of the HeatMap. Then we have a Bipartite graph where vertices are parts and edges are possible connections. Then the problem regarding the search for a matching between vertices known as an assignment problem is

solved. Solution comes from the application of extracted of Part Affinity Fields. Here line integral is computed for all pairs of vertices for a given PAF. Result of the integral gives weight to an edge. Those weights can be referred to as scores and then the problem is to find connections that will produce the maximum total score. Sorting connections and picking the highest one is basically a solution to this assigning problem. Last step is merging. Merging is about converting connections to final human skeletons. It starts with a thought that the number of connections is equal to the number of people. Then the number of people is eliminated by checking the part index of different body parts tuples. Those tuples having the same index correspond to the same part, which leads to the conclusion that they belong to the same human. Procedure is finished when there are no parts with common index. In the output skeleton is produced an array. Each entry is a set containing part coordinates, scores and index.

DeepSort

DeepSort has two targets of application: tracking and detection. Since we handle detection via OpenPose, we omitted this part of the algorithm and used only tracking. Tracking allows us to assign unique IDs to people across frames. In order to achieve it, first of all, DeepSort receives boxes around the person. Those boxes are tuples that look like (x, y, width, high). First two values are rectangle centre and the other two are its width and height respectively. DeepSort takes advantage of this data by using Kalman filters with constant velocity motion and linear observation model. Then Kalman states are generated and the next problem is to create the association between those. Hungarian algorithm handles that using motion information. Tracks are lost if an object does not appear in N number of frames. The value for n can be configured. "Deep" in the algorithm uses CNN based on another metric. This metric in the nutshell is an appearance descriptor. This is a feature vector obtained by eliminating the classification layer and leaving the dense layer, which is responsible for vector generation. Then Mahalanobis distance with this vector as an input completes the job. Two metrics combined demonstrate much better tracking performance than using just the first metric. Actually without a visual appearance descriptor. algorithm does not perform well by having a lot of identity switches when occlusions and noises are present. Output tuples are forwarded to classification models. We use two of those: LSTM, transformer (pictures for both). It has been done for comparison purposes.

For every frame we have 153 are distances between the joints and 153 are angle of joint pairs. We then give those values to classifier.

LSTM

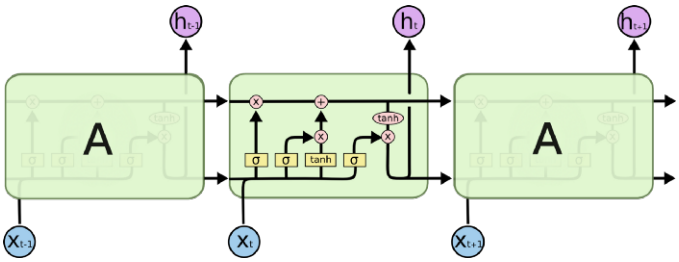

Figure 4. The internal architecture of the LSTM model. Part A is a memory cell. Difference from the model on Figure n is that part A contains memory tools.

Long Short Term Memory (LSTM) algorithm is a type of Recurrent Neural Network (RNN). Similar to RNN, its main feature is feedback connections. It may process not only single data instances but also sequences of data. In the project context it means that not just separate frames passed to the algorithm but the windows of frames. The main problem during work with the time series data is that there may be large gaps of unpredictable length between the two adjacent points in it. LSTM is more tolerant to this problem than traditional RNN. Another main disadvantage of traditional RNN in comparison with LSTM is that it can not deal with the vanishing gradient problem. Vanishing gradient problem occurs during training of models using back-propagation, so back-propagated gradients may vanish. It means that they are tending to zero. It is caused by finite precision numbers used in the computation process, while LSTM allows its units to stay just unchanged in these cases. However, both models are still vulnerable to exploding gradient problems (gradients tend to infinity). LSTM is a technology that takes into account past inputs and preserves them for the amount of time dependent on the weights. In that way, those neural nets are able to store contextual information for a long time. Compared to simple RNNs they make the vanishing gradient problem go away. This problem made weights stop changing their value. Sometimes the outcome was so bad that the neural network could stop from training completely. LSTM in a nutshell consists of recurrently connected memory cells and three units, which are the input, output, and forget gates. Communication between the neural net and cells takes place using gates.

Transformer

Figure 5. Architecture of Transformer. Instead of encoder we use OpenPose and Deepsort. Orange boxes refer to hidden states

Transformer is a deep learning algorithm that was developed in 2017 and introduces the new attention mechanism,

also weights the amount of influence different parts of input data have. Initially it was used for Natural Language Processing (NLP), which is one of the main fields of development of artificial intelligence. We used it for video understanding during our project.

One of the main features of this algorithm is it has a high degree of parallelism of data processing and due to that fact requires much less training time. Low time for training gives an opportunity to use much bigger datasets for training than older popular Recurrent Neural Network (RNN) - Long Short Time Memory (LSTM). We implemented and tested LSTM previously during Project Development to compare results. Attention mechanism and attention layer, corresponding to it, has an access to any state before the current state and assigns weight to each of them according to the level of relevancy to the current state. This process provides more precise information about further states. Attention mechanism is very powerful by itself without RNN and adding it to one of them, for example LSTM, resulted in dramatically increased performance. If we introduce not sequential, but parallel processing of tokens (states) we would get a Transformer. Usually Transformers are based on encoder-decoder architecture. In our case we used Open Pose as an encoder, so we needed only the decoder part of it.

For the testing and training steps we came up with our own dataset.There is the list of requirements for high-quality dataset for DNN-based approach (FCNN): A lot of samples for training and testing (bigger dataset size leads to greater accuracy); An approximately equal number of samples for each action type (increases the precision of process of building probability matrix); The minimum amount of extraneous noise (the performer of the actions should stand out against the background, the background itself should be as plain and simple as possible); Every action must be started and completed on one recording (avoid recordings where the action started on the previous recording or not completed on current one). During our Senior Project, we collected several datasets, each for its own purpose. The first dataset we are using contains video files that we filmed by ourselves and UTKinect-Action3D Dataset. We recorded one of the members of our group performing simple actions. The actions are walking (381 samples), sit down (396 samples), stand up (398 samples), pick up (415 samples), carry (401 samples), throw (407 samples), push (409 samples), wave hands (386 samples), clap hands (419 samples). Every video file lasts 2 seconds. Action names were used for dataset labeling. For every file, only one action had to be performed. We filmed the dataset using the front laptop camera. Group members filmed in the dataset were dressed in dark clothes when the background was completely white and no other objects were in the frame. This was necessary to limit different kinds of noises that could affect the accuracy of our model. However, the resulting dataset was far from the above-mentioned standard. At first we filmed every action for a few minutes and then divided into small 2 second parts. Results were not good because some obtained video files were violating the condition of 2 seconds of an action duration, some of them were longer or shorter in duration than this time. It violated the second requirement from the list. In detail, here are the action categories: some dataset instances had pieces of actions dedicated for another video file, some files were records that represented the action in the middle of a performance, some are just actions where the person was just intended to make an action. This quite accurately falls under the description of requirement number four. This was the result of the application of an algorithm written in Python for an automatic video division. After looking through those results, we decided to divide each video by hand. The final version of the dataset looked much better. The cases described above were significantly reduced. Moreover, certain video files were recorded once again. Based on this data we extracted the human joints and made the preprocessing. One person has 18 joints and the position of each of them has 2 coordinates, x and y correspondingly. This added 18*2=36 new features to the basic 306 features (153 distance and 153 angle features). 342 features in sum. The Fall semester ended at this point. In the Spring semester we had to test a new LSTM algorithm that we implemented to replace the DNN approach. The second dataset had some specific differences from the first one. DNN approach required 2 seconds duration recordings, where the ending of one recording was the start of the next recording. Each recording in the dataset for LSTM was divided into segments with a number of frames equal to 8 with the step of half of the segment (4 frames). Dataset segments in frames: 1-8, 4-12, 8-16, 12-20, 16-24, 20-28, 24-32. Each segment is labeled with the name of an action, which is most frequently present in terms of frames on the segment. Due to this change in the labeling process, the requirements for the qualitative dataset also had slight changes. From this point, there is no need for every action to be started and completed during one recording. In purpose of increasing accuracy we changed some technical properties. First, the number of features was reduced to the basic 306 distance and angle features. Second, implemented scikit-learn python libraries' normalization function. Completion of them had a dramatic impact on the rise of accuracy rate. First semester dataset creation process had struggles, because a big amount of time was spent on the process of choosing the next action to perform during filming. To consider previous mistakes we created a new requirement to change actions very fast and minimize the period between them. This requirement is for LSTM and Transformer to work efficiently, which in result makes accuracy rate higher, which was the problem of the DNN-

based approach. The second dataset is much bigger than the previous one. While the first was around 40 minutes, in summary, the second was more than a 1-hour duration. Also, all segments were labeled by hand. Student filmed in the dataset previously wore black clothes on a white background to minimize side effects. There was approximately an equal number of labels on each action in the table of segments (around 900 labels for each action, 8400 segments total). All requirements were fulfilled with a high degree of accuracy. This resulted in great accuracy of the new algorithm after the testing stage. At the final stage of development of our project, we implemented the Deep Sort algorithm combined with Transformer which added a new opportunity to identify and track multiple persons during one video. The third and the last dataset needed to have a group of people filmed in it. 2 students contributed to the process of production of it. One new requirement added. Persons should do different actions simultaneously. The main problem of the processing of this dataset is that persons covered each other sometimes during the video. That had a big impact on the accuracy. This dataset was merged with the first and the second datasets for training to increase the size of it, and also used randomly chosen 25% of all samples from all datasets for testing purposes. We did it due to the small size of the last dataset. Finally, it is important to say that dataset samples for training are reduced in quality because higher resolution requires more GPU resources. We have been limited in resources, so reducing the resolution was needed to increase the speed of processing.

5. Experimental settings

LSTM architecture:

1. The LSTM model consists of 1 Multi LSTM cell, which consists of 2 Basic LSTM cells.
2. Batch size is equal to 64
3. Decaying learning rate with initial value - 0.0005, decaying rate - 0.96 and decaying steps - 100000
4. 100 epochs.

Transformer architecture:

1. Transformer block contains 8 attention heads (Multi-head attention), 2 normalization layers and 2 dropout layers.
2. There are 4 transformer blocks in architecture, and FCNN (Fully-connected neural network) at the end of the model.
3. Batch-size equals 32.
4. Decaying learning rate with initial value - 0.00005, decaying rate - 0.9 and decaying steps - 100000
5. Number of epochs is 20, as after the approximately 20th epoch there were observed overfitting problems.

6. Results

Single - training, Couple - testing:

In this approach we used our dataset with a single person on recordings as a training set and dataset with a couple persons as testing one.

LSTM:

	precision	recall	f1-score	support
0	0.41	0.52	0.46	83
1	0.39	0.14	0.21	92
2	0.68	0.75	0.72	69
3	0.40	0.10	0.16	79
4	0.47	0.94	0.62	83
5	0.25	0.15	0.19	84
6	0.50	0.15	0.24	84
7	0.42	0.74	0.54	84
8	0.73	0.99	0.84	92
accuracy			0.50	750
macro avg	0.47	0.50	0.44	750
weighted avg	0.47	0.50	0.44	750

Figure 6. LSTM metrics

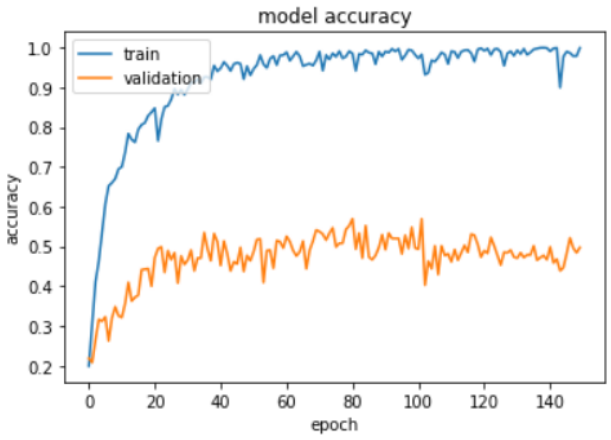

Figure 7. LSTM training

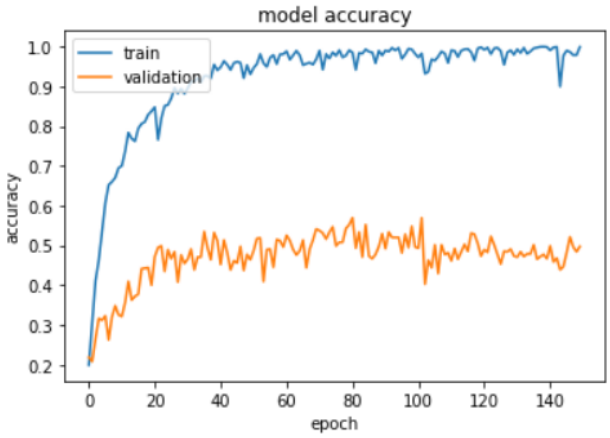

Figure 8. LSTM loss

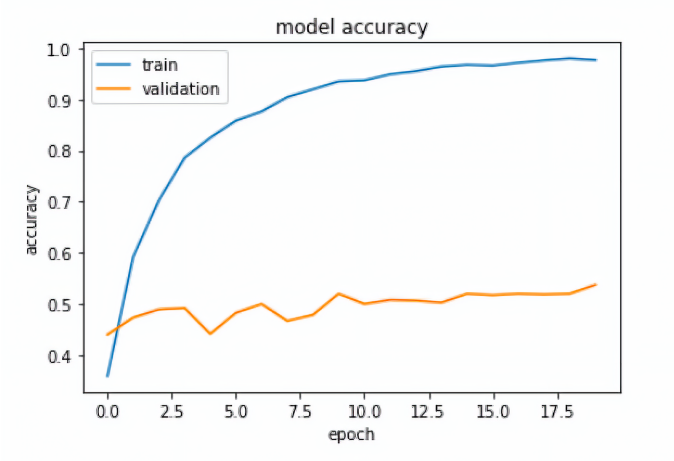

Figure 10. Transformer training

As can be observed from the figures 6, 7, 8,there is quite poor resultant accuracy on the training set. Only about 44% of the testing samples (330/750 samples with sequence of 8 frames) were labeled correctly. Also, the accuracy graph represents that after the 50000th iteration, i.e after approximately the 14th epoch the testing results remain almost the same, however training accuracy growth without overfitting, which is the reason to assign a number of epochs to 100.

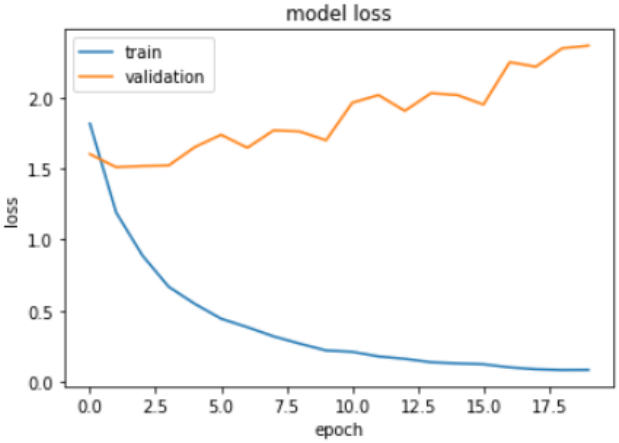

Figure 11. Transformer Loss

Transformer

	precision	recall	f1-score	support
0	0.40	0.46	0.42	83
1	0.67	0.50	0.57	92
2	0.86	0.81	0.84	69
3	0.80	0.20	0.32	79
4	0.84	0.98	0.91	83
5	0.00	0.00	0.00	84
6	0.55	0.14	0.23	84
7	0.28	0.77	0.41	84
8	0.75	0.98	0.85	92
accuracy			0.54	750
macro avg	0.57	0.54	0.51	750
weighted avg	0.57	0.54	0.50	750

Figure 9. Transformer metrics

The transformer architecture has quite better results, compared to the LSTM approach. The final accuracy on the testing set is approximately 54% (405/750 samples with sequence of 8 frames were labeled correctly). Final accuracy is better than in the LSTM approach but still quite low.

Single+Couple; 75% - training, 25% - testing:
After the "dataset with single person - training and dataset with couple persons - testing" division approach we decided to merge them together to see how much results will improve. 25% of the merged dataset was used for validation purposes.

LSTM:

	precision	recall	f1-score	support
0	0.97	0.97	0.97	117
1	0.97	0.93	0.95	119
2	0.98	0.99	0.99	106
3	0.98	0.93	0.96	126
4	1.00	1.00	1.00	118
5	1.00	0.99	1.00	114
6	0.97	1.00	0.98	116
7	0.95	0.97	0.96	112
8	0.97	1.00	0.99	115
accuracy			0.98	1043
macro avg	0.98	0.98	0.98	1043
weighted avg	0.98	0.98	0.98	1043

Figure 12. LSTM metrics for 75% trainig 25% testing

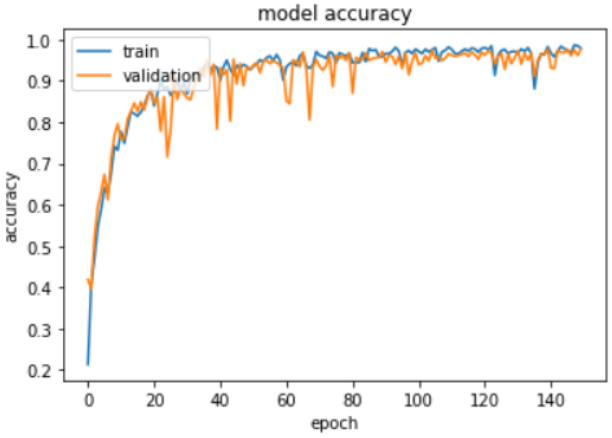

Figure 13. LSTM training for 75% trainig 25% testing

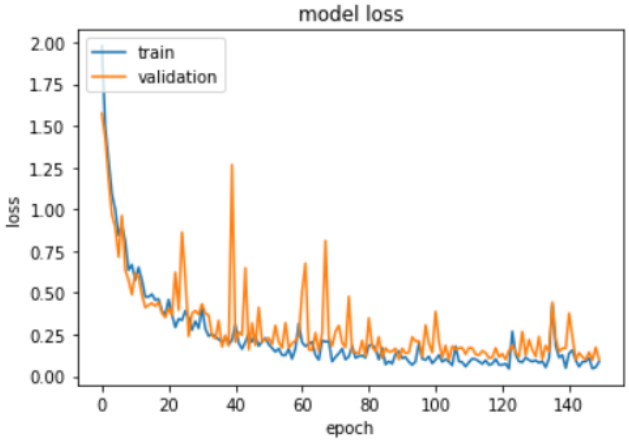

Figure 14. LSTM loss for 75% trainig 25% testing

As can be observed from the above graphs, the improvement of LSTM approach was essential. Accuracy at the testing set growed up from 44% to 97.5%.

Transformer:

	precision	recall	f1-score	support
0	0.95	0.97	0.96	117
1	0.97	0.95	0.96	119
2	1.00	0.97	0.99	106
3	0.98	0.95	0.97	126
4	0.99	1.00	1.00	118
5	0.98	1.00	0.99	114
6	0.97	0.98	0.97	116
7	0.96	0.98	0.97	112
8	1.00	1.00	1.00	115
accuracy			0.98	1043
macro avg	0.98	0.98	0.98	1043
weighted avg	0.98	0.98	0.98	1043

Figure 15. Transformer metrics 75 % training 25 % testing

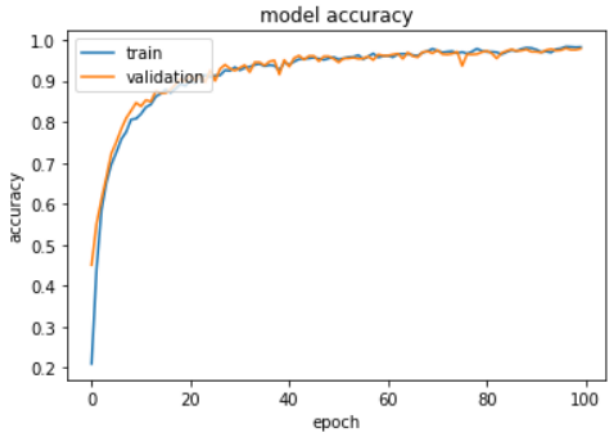

Figure 16. Transformer training 75 % training 25 % testing

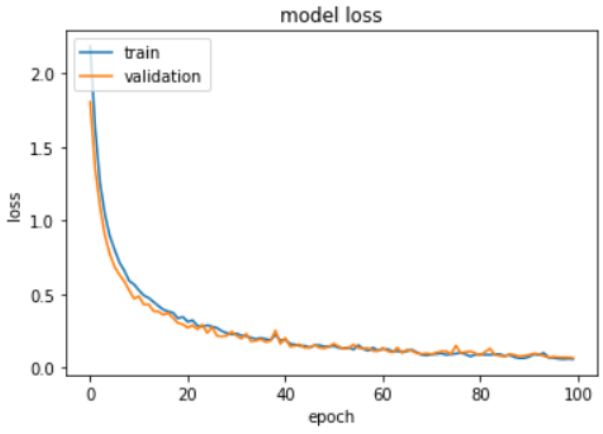

Figure 17. Transformer loss 75 % training 25 % testing

The improvement of the Transformer approach was also quite essential. Accuracy on the testing set growed up from 54% to 98% which is the best result over all approaches.

Single+Couple(75%) - training, only Couple dataset - testing:

Transformer

	precision	recall	f1-score	support
0	0.89	0.99	0.94	83
1	0.96	0.86	0.91	92
2	1.00	0.97	0.99	69
3	0.89	0.84	0.86	79
4	0.98	0.98	0.98	83
5	0.98	1.00	0.99	84
6	0.85	0.94	0.89	84
7	0.99	0.95	0.97	84
8	1.00	1.00	1.00	92
accuracy			0.95	750
macro avg	0.95	0.95	0.95	750
weighted avg	0.95	0.95	0.95	750

Figure 18. Transformer metrics for 75% training and couple testing

We also decided to test our trained model with a merged dataset (75% - training, 25% - testing) approach to test it only the dataset with a couple persons. The accuracy is above 96 percent, which is also quite good.

Discussion

Based on the above results, we can conclude that the models show a rather low result when tested on an unobserved dataset. Despite the fact that the actions are the same, many factors are quite different in a dataset with one person and in a dataset with a couple of people. First, the data is very imperfect. In many frames, a person is preparing to perform an action, for instance, there is a large number of frames where a person just stands, but they are labeled as another action. Despite the fact that during training the model can adjust to these inaccuracies, they are clearly visible on testing with a different dataset. Secondly, the efficiency of the OpenPose algorithm. The fact is that with an increase in the number of people, its accuracy drops due to a large number of different noises that affect the detection of joints. Also, as a rule, the more people in the frame, the lower resolution they are displayed on this frame. Third, there are a lot of flaws in the dataset. The main overlap is when one person overlaps some important joints of another, and therefore the number of dropped frames was quite large, almost 10% of the total number of frames from the entire dataset with a couple of people. Fourth, a slightly different background on which the recording was made. This problem is not so significant, but it also affects the final result. Below is a confusion matrix for this approach.

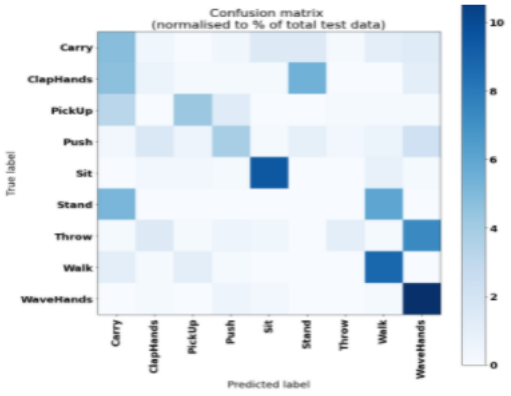

Figure 19. Confusion matrix

As it can be observed, the algorithm is confused in actions that, when disassembled frame-by-frame, are very similar to each other, for example, ClapHands-Carry, Stand-Carry, Throw-WaveHands and so on.

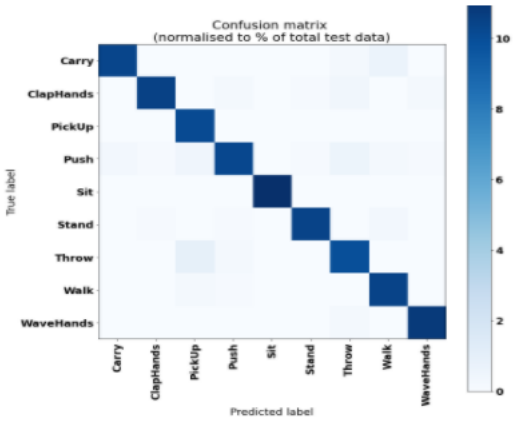

Figure 20. Confusion Merged

Now, if we look at the confusion matrix with the merged dataset, we can see that it adjusts for the difference in conditions and many of the problems described above will no longer strongly affect the final testing accuracy. Almost all actions are recognized quite accurately. Even when testing the finished model only on a dataset with a couple of people, it expectantly shows quite good results.

7. Conclusion

During our research, we tested the results using various algorithms and compared their results. The main criterion for evaluating ultimately is the accuracy of the model. From the above results, it follows that, in general, the Transformer model shows a significantly higher result (5-10% on aver-

age), which is a serious indicator. Comparing different approaches with respect to datasets, as expected, when samples from a dataset with several people are part of a training dataset, it shows much higher results. This is because it adjusts its variables for both datasets and is already able to recognize actions more flexibly. Also important is the fact that some of the actions we have chosen have very similar dynamics, which greatly affects the accuracy of the model on a never-seen dataset. Also it should be taken into account that our datasets are quite small in order to train such complex models. Despite all of the above, the results of the dynamic approach (LSTM and Transformer) generally show better results in comparison with the static approach (FCNN and CNN).

Future studies:

1) Optimization of models: accelerating the processing speed, testing various approaches (for example, using only every 2 or 3 frames for processing), adapting to work in real time, selecting lighter models (especially OpenPose), etc.
2) Selection of more different actions for our datasets, since some of the current ones have very similar dynamics, which affects the accuracy.
3) Collecting a better and much larger dataset
4) Search and use of other features that potentially carry more information that we are using now, expanding our vector of features.
5) Increasing accuracy: testing using various types of data normalization, testing using noise filters, etc.

References

[1] F. Angelini, Z. Fu, Y. Long, L. Shao, and S. M. Naqvi, "2d pose-based real-time human action recognition with occlusion-handling," *IEEE Transactions on Multimedia*, vol. 22, no. 6, pp. 1433–1446, 2019.

[2] F. Chen, "Real-time action recognition based on human skeleton in video," 2019.

[3] C. Gu, C. Sun, D. A. Ross, C. Vondrick, C. Pantofaru, Y. Li, S. Vijayanarasimhan, G. Toderici, S. Ricco, R. Sukthankar, *et al.*, "Ava: A video dataset of spatio-temporally localized atomic visual actions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6047–6056.

[4] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "Mot16: A benchmark for multi-object tracking," *arXiv preprint arXiv:1603.00831*, 2016.

[5] D. Neimark, O. Bar, M. Zohar, and D. Asselmann, "Video transformer network," *arXiv preprint arXiv:2102.00719*, 2021.

[6] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3645–3649.