CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

F3

Faculty of Electrical Engineering
Department of Cybernetics

Bachelor's Thesis

Dance Genre Recognition from a Video of a Dancing Pair

Štěpán Křivánek
Open informatics

August 2022
Supervisor: prof. Ing. Jiří Matas, Ph.D.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Křivánek Štěpán** Personal ID number: **492179**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Dance Genre Recognition from a Video of a Dancing Pair

Bachelor’s thesis title in Czech:

Rozpoznávání tanečního žánru z videozáznamu páru

Guidelines:

1. Propose an approach for ballroom dance recognition that is based on visual information only. The method will be tested on videos with one or more dancing pairs.
2. Collect a suitable set of data for both training and testing. Consider both manual annotation and noisy annotation by the audio-based program for dance recognition developed by T. Pavlín [1] and the dataset of of P. Kouba [2]
3. Solve the problem of tracking of individual dancers using a representation of skeleton keypoints and their clustering into dancing pairs.
4. Apply a selected method for activity recognition, using the tracks of the pairs. The method should aim at recognising the dance from the broadest set of viewpoints.
5. Evaluate the method on tracks of varying length and estimate the time necessary to reliably estimate the dance category. Compare your results with the work of P. Kouba [2]

Bibliography / sources:

[1] Tomáš Pavlín: Dance Recognition from Audio Recordings (Master's thesis, 2020).
[2] Petr Kouba: Recognition of Dance Genres from Video (Master's thesis, 2021).
[3] Z. Cao and G. Hidalgo Martinez and T. Simon and S. Wei and Y. A. Sheikh : OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields (IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019).

Name and workplace of bachelor’s thesis supervisor:

prof. Ing. Jiří Matas, Ph.D. Visual Recognition Group, FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **03.02.2022** Deadline for bachelor thesis submission: **15.08.2022**

Assignment valid until: **30.09.2023**

prof. Ing. Jiří Matas, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt	Student’s signature

Acknowledgement / Declaration

I would like to thank prof. J. Matas for his patience and motivation, when it was needed the most, as well as Petr Kouba for providing me with his previous work (including valuable datasets) and also for his willingness and fresh ideas.

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 15 August 2022

.....

Abstrakt / Abstract

Navrhujeme vylepšení nedávné práce Petra Kouby, která se týká rozpoznávání společenských tanců z videa. Soustředili jsme se výhrádně na vizuální data, a abychom mohli naše výsledky porovnat, použili jsme stejné datasety. Stejně jako Petr Kouba jsme tanečníky reprezentovali jako skeletony. Nicméně jsme pro jejich extrakci nahradili OpenPose za HRNet pro dosažení větší přesnosti skeletonu a MS-G3D jsme zaměnili za PoseC3D pro přesnější klasifikaci tanečních žánrů. Přesnost (Top-1) nově navrhované metody dosahuje na 10-Let's Dance datasetu 77.4%, což je přibližně o 22% více než metoda navrhnutá v práci Petra Kouby, ta dosahuje 55.5%. Dále jsme metodu otestovali na datasetu Dance Tutorials Dataset, kde jsme dosáhli přesnosti 74.5%, což je asi o 10% více (64.8%).

Klíčová slova: Tanec, Rozpoznávání , Klasifikace, HRNet, PoseC3D, OpenPose, MS-G3D, ByteTrack

Překlad titulu: Rozpoznávání tanečního žánru z videozáznamu páru

We propose improvements of the recent work of Petr Kouba, concerning ballroom dance genre recognition from video. We focused on visual data only and for comparability we use the very same datasets. Similarly to Petr Kouba we use a skeleton representation of the dancers. However, we replaced OpenPose with HRNet for pose estimation to achieve better precision and MS-G3D with PoseC3D for more accurate classification of the dance genres. The Top-1 accuracy of our newly proposed method evaluated on the 10-Let's Dance dataset is 77.4%, which is approximately 22% higher than the Top-1 accuracy of the method proposed by Petr Kouba (55.5%) and also on the Dance Tutorials Dataset achieving the Top-1 accuracy 74.5%, which is roughly 10% higher (64.8%).

Keywords: Dance, Recognition, Classification, HRNet, PoseC3D, OpenPose, MS-G3D, ByteTrack

Contents /

Tables / Figures

Chapter 1
Introduction

Dance genre recognition is still quite an unexplored field as it is pretty specific. When it comes to dance, probably everyone would at some point think about music, since these two come together just like a fork with a knife. Furthermore, dance genres are often in correlation with music genres, therefore one may assume a dance genre from both, music or dancer's movements.

Although it is probably a bit more natural to approach such problem as the dance genre recognition by audio based methods, the audio may not always be available. Such methods could also suffer from too much ambient noise. In this thesis we focus on the visual perception only. However, there are still many ways one can handle the visual information given. One could for example try to classify the action performed just by the flow of the image pixels, that would likely come with a problem to separate the ambience, even tho the information about ambience could in some cases be also helpful. To get rid of this problem we first separate the human skeletons and then work just with them. In this thesis we study the application of the state of the art skeleton based action recognition method to dance genre recognition and compare it to another similar work.

Two recent theses [2] and [1] look respectively into audio and visual based dance genre recognition. Since we examine the visual based recognition only, the thesis of Petr Kouba [1] is at our most interest. As the field of computer vision and science in general still grows quickly, newer and better tools have already been developed and released. Therefore there are ways to further improve the dance genre recognition method proposed in the thesis [1] and in this thesis we will introduce a new model (involving pose estimation, tracking and action recognition) for visual based dance genre recognition. The new model preserves the idea to interpret people, who are the subject of our interest, as skeletons with keypoints, but improves most of the core segments of the model. All these new segments are a part of the OpenMMLab [3] project, which includes various benchmarks and toolboxes with implementations of the current state of the art methods in action recognition.

Considering dance genres, there is way too many of them and both theses work with a set of data, which consists of ballroom dances belonging to the so-called International Style. Ballroom dances involve 5 standard dances and 5 Latin dances as shown in the Table 1.1. In the thesis [1] a term 'The 10 Dances' is introduced to simplify nomenclature and so we will follow the convention and use it as well for any further mentions of these dance genres. For a clear comparison of the new model we will limit ourselves only to the The 10 Dances as well and we will also train and validate our model on the very same datasets as used in [1].

Latin dances	Standard dances
Rumba	Tango
Cha-Cha	Slowfox
Jive	Quickstep
Paso Doble	Viennese Waltz
Samba	Waltz

Table 1.1. 'The 10 Dances' belonging to the so-called International Style. The table has been taken from [1].

Chapter 2
Datasets

An integral part of any machine learning are datasets, as a lot of data is usually needed for a model to learn enough to perform well on the real-world data. In this chapter we will provide information about the datasets used throughout our experiments and also about those which were important for the new dance genre recognition model. Since this work is very closely related to the one of Petr Kouba [1] and we want our results to be well comparable with his ones, we will be using the very same datasets and their modifications.

2.1 The base datasets

The base of our datasets consists exactly of those datasets used in the thesis [1]. That involves three datasets - Dance Tutorials Dataset, 10-Let's Dance, and YT8M Ballroom. However, we used only the first two of them for proper training and validation. The YT8M Ballroom dataset was in the thesis [1] mostly used only to enhance the Dance Tutorials Dataset with more, yet noisy data. However as the results of [1] show, there was no significant improvement, and thus we decided to omit it from our tests and rather focus on the other datasets.

2.1.1 Dance Tutorials Dataset

The Dance Tutorials Dataset was manually obtained and annotated by Petr Kouba as a part of his thesis [1]. This dataset is constructed to involve a single dancing pair per video with no other people in the background and it includes exactly the genres from The 10 Dances. Therefore it should be an example of a best case scenario dataset. However, the dataset is quite small as it contains only 13 - 21 unique videos per genre with a total footage of roughly 10 minutes per genre, the exact distribution of this dataset is shown in the Table 2.1. Due to the low variety of unique videos the dataset is likely to be easily overfitted. Moreover, the dataset includes videos with very different lengths, which may cause problems for the training, as the dance genre recognition model might focus on improving the genres with more footage at the expense of others.

Fortunately the thesis [1] provides us also with a manual annotation on how to split the data into training and validation ones to preserve roughly the same ratio for all genres. This also means we can compare the models trained on the very same data. In this thesis we will proceed to compare both, manually annotated as well as randomly split data.

2.1.2 10-Let's Dance

The 10-Let's Dance dataset is again introduced in the thesis [1]. Nine of the ten genres originate from a Let's Dance dataset [4], however since Viennese Waltz is missing in this dataset, it was added later as a part of the thesis [1]. Therefore one should bare in mind, that some inconsistencies regarding Viennese Waltz in this dataset can result from possibly too different samples in comparison to the other genres. Compared to

Genre	Total footage	Unique videos
Cha-Cha	10:10	13
Slowfox	10:19	16
Jive	10:27	15
Samba	10:23	16
Tango	10:21	16
Waltz	10:20	18
Paso Doble	10:23	17
Quickstep	10:19	21
Rumba	10:25	16
Viennese Waltz	10:24	21

Table 2.1. Classes in Dance Tutorials Dataset, total footage of their instances and number of different videos from which the instances were obtained. The table has been taken from [1].

the Dance Tutorials Dataset, The 10-Let's Dance dataset includes more variant video instances with slightly longer footage per genre. Also the instances do not differ as much in their length, so there is no problem in splitting the dataset just at random. However, that also means that our training and validation datasets do not exactly match those used in [1].

Another major difference is that the videos from this dataset usually contain multiple people. They usually include more than a single dancing pair and also often many spectators, which are not involved in the action performed to be recognized. That sets us a challenge to distinguish the relevant people from the misleading ones. Unlike the Dance Tutorials Dataset, which consists from dancing tutorials as its name suggests, the 10-Let's Dance dataset is mostly made out of videos capturing a dance competition. That means it includes more complex and professional dance moves instead of just the basic ones.

Genre	Total footage	Number of instances
Cha-Cha	16:04	96
Slowfox	13:03	77
Jive	17:44	102
Samba	15:59	94
Tango	13:13	79
Waltz	13:23	79
Paso Doble	16:04	94
Quickstep	13:23	80
Rumba	15:14	91
Viennese Waltz	13:31	80

Table 2.2. Classes in 10-Let's Dance, total footage of their instances and number of instances per class. Some of the instances from 10-Let's Dance were not available due to territorial copyright restrictions or due to the video being removed from Youtube. The table has been taken from [1].

2.1.3 YT8M Ballroom

YT8M Ballroom is the last dataset used for experiments in the thesis [1]. As already mentioned, the purpose of this dataset was to provide more data for the Dance Tutorials Dataset and eventually for the 10-Let's Dance as well. However, it turned out to make no notable difference, and thus we will omit this kind of experiments and will limit ourselves to the other two datasets.

2.2 Datasets used for pre-training the models

There also is a few datasets, which we did not use directly during our training or testing, but they were used for pre-training various parts of the model and therefore they deserve to be mentioned as well. Most of them will probably be well known to everyone, who has ever gotten in touch with the action recognition before.

2.2.1 COCO

COCO [5] is a large scale dataset for object detection, segmentation and key-point detection amongst others. It includes predefined training, validation and testing datasets, which altogether consist of 250000 human images with labeled keypoints. Therefore is widely used for training the keypoint detection.

2.2.2 Kinetics-400

Kinetics-400 [6] is the original version of the Kinetics dataset, which contains 400 different human action classes with variant YouTube video clips. It is a pioneer between the datasets aiming for human action recognition. Later, extended versions of this dataset with 600 and 700 classes were released.

2.2.3 FineGym

FineGym [7] is another action recognition dataset. However, as the name may suggest, all of the videos from this dataset come from various disciplines performed in a gymnasium. This dataset also contains people, who do not belong to the action performed, such as audience and referees, and involves very closely related classes.

2.2.4 NTU-60

NTU-60 [8], originally called NTU RGB+D, is a large-scale dataset for human action recognition captured by a depth camera and therefore allowing for 3D human action recognition.

Chapter 3
State of the art

The field of computer science and the computer vision in particular progresses at the speed of light and what was considered as the state of the art yesterday may not be so today anymore. Despite the fact Petr Kouba published his thesis just an year ago, basically all of the parts of his model are nowadays outdated. Many improvements have been made in all three major subjects of interest for skeleton-based action recognition, that involves pose estimation, pose tracking and finally the action recognition itself built on top of these. In this chapter we will discuss their improvement evaluated on the most relevant benchmarks for our task of dance genre recognition.

3.1 OpenMMLab project

First let's take a look at the OpenMMLab project [3] as it turned out to be at our most relevance. Briefly, it is a project which includes and unifies a lot of open-source projects from the field of computer vision and deep learning. That includes our subjects of interest, as the contributors of this project are active researchers from all over the world in the field of action recognition. OpenMMLab includes a lot of different toolboxes, but the most important for us are the ones for pose detection MMDetection [9], pose estimation MMPose [10], object tracking MMTracking [11] and last but not least a toolbox for general video understanding, which involves action recognition, MMAction2 [12]. All these toolboxes together create a complete set of what one may need for dance genre recognition. Moreover they are all well documented and open-source.

3.2 Human detection

Our first point of interest is the human detection. There are many methods, which aim at detecting various objects and that usually includes human as well and therefore they would suit our task. As we want the model to have wide range of possibilities to be used for, we decided to focus mainly on methods for real-time detection. One of such widely used methods is Faster R-CNN [13], which was also used in the state of the art model for skeleton based action recognition [14]. And therefore it deserves our attention. However, as the benachmarks [15] show, the current state of the art for real-time object detection are modifications of the YOLO [16] series detectors. Some of them are also implemented in the MMDetection [9] toolbox.

3.3 Human pose estimation

Our second point of interest is the human pose estimation. As mentioned in [1] and according to the state of the art pose estimation benchmark [17], OpenPose, which was used for the pose estimation of dancers in [1], has already fallen behind the current state of the art a lot. There are many more methods, which achieve better results nowadays,

but we focused in particular on HRNet as it is again the method used in the current state of the art model for skeleton based action recognition [14]. This model and its derivations have currently been used by many projects leading human pose estimation as well as multi-person [18] pose estimation benchmarks. Just to mention a few, they achieve great results on MPII Human Pose [17] and COCO [19]. And again there is also an implementation of HRNet in MMPose [10], which in addition also claims to achieve the same results as other HRNet implementations at higher speed. See their benchmark[1] tested on COCO 2.2.1 dataset.

3.3.1 Top-Down vs Bottom-Up approach

Since we want to compare HRNet [19] with OpenPose [20], we first need to understand the difference of their pose estimation approach. In general the Top-Down approach first detects all people on a given image and sets tight bounding-boxes around them, then it proceeds to detect the keypoints of each individual person by a single-person pose estimator. The disadvantage of this approach lies in its speed, as it needs to process each image for every person detected. Yet if there are very few people in the image, the Top-Down approach can still be comparably fast to the Bottom-Up.

On the other hand, the Bottom-Up approach first detects all keypoints on a given image and then tries to connect these keypoints into human poses or at least their parts. The first part (detection of the keypoints) of this approach is usually quite fast, but the second part may struggle to connect those keypoints and can take a while. A work [20] proposes a method known as OpenPose, which can estimate human poses at real-time. However, it also comes with a greater risk of the keypoints not being correctly connected, which can affect the neighboring frames in further usage as well.

OpenPose is probably the best known human pose estimator using the Bottom-Up approach and was used in the thesis [1], however most of the state of the art models nowadays prefer the Top-Down approach. Nevertheless, the Bottom-Up approach may still perform better at some specific tasks, e.g. to perform a real-time human pose estimation from a mobile phone camera. MMPose [10] does not include an implementation of OpenPose, but it comes with an implementation of both approaches for HRNet [19]. According to the state of the art in skeleton based action recognition [14], the Top-Down HRNet performs far better than its Bottom-Up sibling on the COCO 2.2.1 dataset and slightly better on the NTU-60 2.2.4 dataset as well. The results achieved in this work are shown in the Table 3.1.

Pose Estimator	COCO AP	NTU-60
HRNet (Top-Down)	**0.746**	**93.6**
HRNet (Bottom-Up)	0.654	93.0

Table 3.1. Comparison of HRNet-w32 Top-Down and Bottom-Up approach on COCO and NTU-60. The table has been taken from [14].

3.4 Tracking

Our third point of interest is the tracking of the human poses. Since human pose estimation provides us only with skeletons on each frame, we also need to somehow keep the track of which skeleton belongs to which one in the continues frames. Considering

[1] `https://github.com/open-mmlab/mmpose/blob/master/docs/en/benchmark.md`

dancing, it becomes quite a difficult task as the dancers keep changing positions quickly, overlap each other from the camera's point of view and can also partially or even completely disappear from the view. There can also be more people in the video, which in our case includes groups of pairs or people who are not dancing and may therefore contribute as a noise.

3.4.1 HRNet tracking performance

The work [19] also proposes a pose tracking method for their pose estimation network. As stated in the chapter 4.3 in [19], the tracking method is based on the one proposed in SimpleBaselines [21]. The method combines the bounding boxes obtained from a single image object detector with those calculated from neighboring images by the optical flow of the video. When the bounding boxes are combined, they apply a non-maximum suppression to reduce the greatly overlapping bounding boxes into the one with the highest confidence.

The work [19] compares HRNet with the tracking method described above with other models on the PoseTrack2017 dataset as shown in the Table 3.2. The metrics used for the comparison are mAP, which is the mean average precision calculated throughout the estimated poses, and MOTA, which stands for multi object tracking accuracy and takes into account the object detection as well as tracking . For more details about these metrics, follow the work [19]. At the time of its publication, HRNet was dominating the field of human pose tracking. Few years passed since then and according to [22] only very few new models managed to overtake the HRNet on PoseTrack2017 dataset considering the MOTA metric and none considering the mAP metric.

Entry	Additional training Data	mAP	MOTA
ML-LAB	COCO+MPII-Pose	70.3	41.8
SOPT-PT	COCO+MPII-Pose	58.2	42.0
BUTD2	COCO	59.2	50.6
MVIG	COCO+MPII-Pose	63.2	50.7
PoseFlow	COCO+MPII-Pose	63.0	51.0
ProTracker	COCO	59.6	51.8
HMPT	COCO+MPII-Pose	63.7	51.9
JointFlow	COCO	63.6	53.1
STAF	COCO+MPII-Pose	70.3	53.8
MIPAL	COCO	68.8	54.5
FlowTrack	COCO	74.6	57.8
HRNet-W48	COCO	**74.9**	**57.9**

Table 3.2. Results of pose tracking on the PoseTrack2017 test set. The table has been taken from [19].

3.4.2 Current state of the art tracking methods

As the HRNet article [19] was published in 2019 (which may not seem too long ago, but as we already saw it kind of is in this field), the tracking methods were improved since then as well. The datasets we use always consist out of at least two people (the dancing pair) and often more, therefore we explore the multiple object tracking methods. Probably the most prestigious benchmarks for multiple object tracking are nowadays those published in MOT Challenge [23]. If we take a look at the MOT17 [24] and

MOT20 cite[MOT20] benchmarks, the currently leading methods considering MOTA score are BoT-SORT [25] and ByteTrack [26].

However, there is also another very relevant benchmark for our purposes. The benchmark [27] evaluates the multiple object tracking on the DanceTrack dataset. As the name suggests the DanceTrack dataset consists of various dance genre videos. These genres are not quiet the same as The 10 Dances and therefore we do not use this dataset in this thesis. Nonetheless, there is no information about the performance of the BoT-SORT method in the benchmark [27], however the ByteTrack method ranks yet again second, considering the MOTA score. Since this benchmark is evaluates tracking of dancers, we can expect the ByteTrack to have very similar performance on the Dance Tutorials Dataset and 10-Let's Dance dataset. Another benefit of ByteTrack is that there is also an implementation of it in MMTracking [11], therefore we decided to further inspect ByteTrack in our experiments.

3.5 Action recognition

Our last point of interest is the action recognition. Now when we are finally able to extract and properly track the skeletons, we would like to estimate the action performed by them, which in our case is the genre of the dance performed by the skeletons in a given video.

3.5.1 Graph convolutional networks

Nowadays the most dominant methods for action recognition are based on the graph convolutional networks (GCN). The basic idea of these methods is to model skeleton sequences as a spatiotemporal graphs. Such graph consists of a set of vertices, which represent the keypoints of the skeletons, and a set of edges, which connect the keypoints to form a human pose (just like bones) and also connect the same keypoints throughout the consecutive frames. After building such graph, they perform convolution just like convolution neural networks (CNN), but generalized to be able to work with any number of neighbors per each cell (vertex in case of GCN).

Petr Kouba in his work [1] already discusses the state of the art of GCN-based methods. The state of the art of these methods did not change too much and therefore please follow his work for more information about it. He also in his work adopts one of the highest scoring GCN-based methods on Kinetics-Skeleton dataset [28] called MS-G3D, and thus it will be of our interest.

3.5.2 Convolutional neural networks

In spite of the stagnation of GCN-based methods, the work [14] proposes another different approach for skeleton-based action recognition and proposes a new model called PoseC3D. This model now ranks first at the Kinetics-Skeleton dataset [28] and therefore it naturally caught our attention. PoseC3D achieves the accuracy of 49.1% on the Kinetics-Skeleton dataset [28]. This may not seem as much of the first sight, but it is actually improvement by more than 10% in comparison to the best GCN-based methods (MS-G3D scores 38.0%).

PoseC3D follows on from another approach for skeleton based action recognition, which adopts classical 2D-CNN or 3D-CNN. As mentioned in the work [14], PoseC3D transforms each skeleton into a heatmap by calculating a gaussian heatmap centered at each keypoint for each frame. If there are more skeletons in a frame, their heatmaps get accumulated. When each frame is processed like this, all frames are stacked together

by the temporal axis and therefore form a 3D volume heatmap. This heatmap is then fed into the 3D -CNN. The whole process and architecture of the PoseC3D network are depict in Figure3.1.

The work [14] argues that the reason for using a 3D-CNN instead of a 2D-CNN is because of its good capability for spatiotemporal feature learning. According to [14], the GCN-based methods lack robustness and scalability. Moreover, since PoseC3D can just accumulate the heatmaps of multiple skeletons, the input preserves the same size regardless of the number of skeletons, which is not the case of GCN-based methods as their graphs need to be extended.

Besides, PoseC3D is currently at the top of the ranks of many other action recognition benchmarks [29], including the skeleton-based action recognition. Moreover, there is also its implementation included in MMAction2 [12], which also makes it a great candidate to fulfill the last part the model for skeleton-based dance genre recognition.

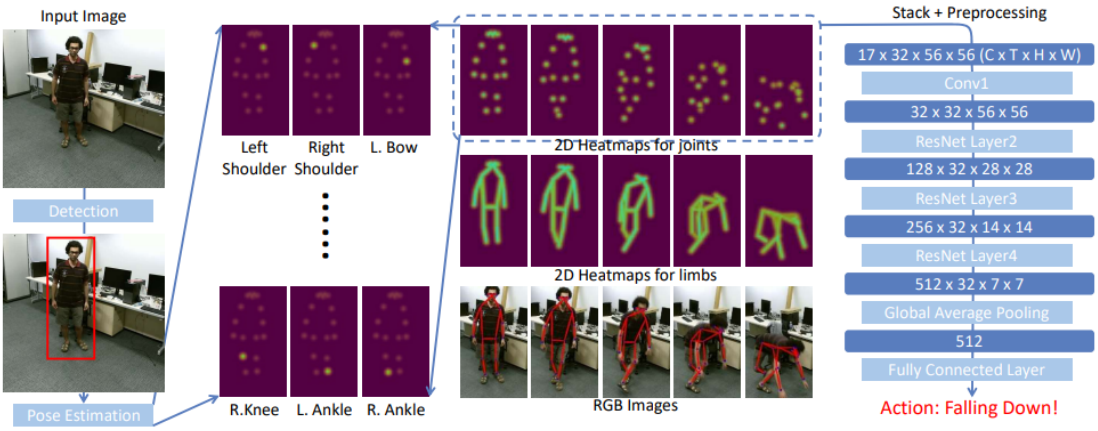

Figure 3.1. Schema of the PoseC3D model. The figure has been taken from [14].

Chapter 4
Objects of interest

In this chapter we describe all the parts of the observed pipeline for visual based dance genre recognition. We observe the current state of the art method for skeleton based action recognition PoseC3D and its application to dance genre recognition. We also combine this method with the method described in the thesis of Petr Kouba [1] to see which of the parts have the greatest impact on the gained results.

We then provide a diagram of the pipelines in the Figure 4.2. Notice, that we also feed PoseC3D network with skeletons without previous pair detection and without removing the audience. Since PoseC3D handles well even large amount of human skeletons, we do not need to reduce the amount of detected skeletons and observe its impact. We further refer to this as No-Reduction method.

4.1 Pose estimation

The work [14] adopts the Top-Down approach of HRNet-w32 for pose estimation. Since the benchmarks presented in the section 3.3 and 3.4 have shown that HRNet performs quiet well in both, pose estimation and tracking, we decided to stick with it as it should be a strong competitor for OpenPose. However, we also follow the proposed Top-Down approach of HRNet as it scores better. This causes some incompatibilities discussed later in the section, because OpenPose implements a Bottom-Up approach. We discuss this problem and its solution later in the section 4.3.

HRNet also uses a slightly different format of the skeletons than OpenPose as depict in the Figure 4.1. There are multiple formats of the skeleton for OpenPose., a 25-keypoint one and 18-keypoint one. As recommended in the thesis [1] we use the 25-keypoint one for a better precision and then drop the feet keypoints to obtain the 18-keypoint skeleton. HRNet skeleton on the other hand uses a 17-keypoint format. To transform the OpenPose skeleton to HRNet format we drop the neck keypoint and reorder the rest. To transform the HRNet skeleton to the OpenPose format we calculate the missing neck keypoint as the average of the two shoulder keypoints, that includes the confidence calculation.

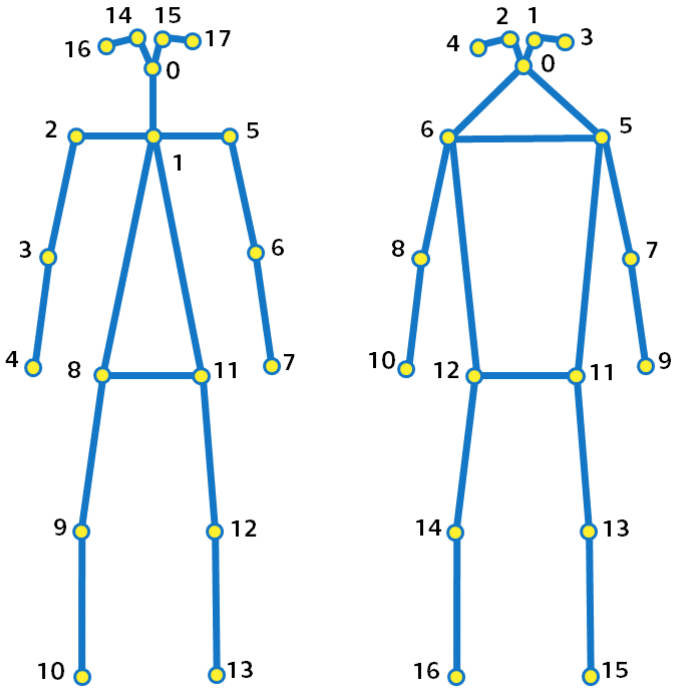

Figure 4.1. OpenPose and HRNet skeleton comparison. OpenPose skeleton depict on the left and HRNet on the right.

4.2 Human detection

Considering we use a Top-Down pose estimation approach, we first need to select a detector, which provides us with bounding boxes of detected humans for each frame. The work [14] adopts for this purpose a widely used Faster R-CNN [13]. However, the method we want to use for tracking 4.3 prefers other detector called YOLOX [30]. We can notice that it is a modification of a the detector discussed in the section 3.2. Moreover we can see on the benchmarks [31], that it is a highly ranked method at various real-time object detection. For these reasons we decided to go with the YOLOX method.

4.3 Human tracking

In the task of dance genre recognition, it may be also appropriate to track the detected humans, so we can treat each human consistently throughout all the frames. For our purposes, this is a very important prerequisite for a successful reduction of the crowded video into a single dancing pair. This procedure is further discussed in section 4.4. As the results of section 3.4 show, the most suitable tracking method for the dance genre recognition currently seems to be ByteTrack [26] and so we will adopt it into our pipeline.

Just like the other currently best tracking methods, ByteTrack [26] aims to track the bounding boxes rather than the skeletons themselves afterwards. However, since OpenPose is a Bottom-Up approach pose estimator, it does not provide us with bounding boxes. To create them with calculate the minimum and maximum of the x and y coordinates of keypoints for each skeleton. We would now like to feed the obtained

bounding boxes into ByteTrack. However the implementation we use has closely connected YOLOX detector to ByteTrack and we did not manage to feed ByteTrack with the bounding boxes from OpenPose.

For that reason we figured another way to set the track ids to the bounding boxes from OpenPose. We first run ByteTrack with YOLOX and get the bounding boxes with track ids. Then we proceed to calculate the intersection over union (IoU) of each of the bounding boxes with each of those from OpenPose. Then we select the pair of bounding boxes with the highest IoU score, pass the track id ByteTrack bounding box to OpenPose bounding box and and remove these two bounding boxes from this procedure. We run this for each frame separately and repeat until either all ByteTrack bounding boxes passed their track id or all OpenPose bounding boxes were assigned a track id. The OpenPose bounding boxes with no track id are later in the pair reduction procedure 4.4 thrown away. Therefore bear in mind that the experiments with OpenPose skeletons going through ByteTrack-Reduction described in section 4.4 are laden with a greater error caused by this method.

For the comparison, Petr Kouba in his work [1] groups the skeletons in consecutive frames based on their pose. His method matches the skeletons in consecutive frames based on their keypoints distance, the lower the distance the better. Then he applies the same greedy method. He selects the best pair and removes it from the selection, but minimizes the keypoint distance instead of maximizing the IoU of the bounding boxes.

4.4 Pair reduction

Now that we have a set of skeletons, which are all assigned a track id, we connect the skeletons throughout the frames based on their track id. Then we need to reduce them into a single dancing pair. To achieve this we first try to match all skeleton into pairs. That is not a difficult task considering the Dance Tutorials Dataset as there usually already is just a single pair. However, it is a very challenging task on the 10-Let's Dance dataset, because it consists of multiple dancing pairs and a lot of other people in the audience. When we have the skeletons connected into the pairs, we select the best dancing pair, while trying lower the chance that a pair in audience gets selected. We further refer to this pair reduction as ByteTrack-Reduction, since it relies on the tracked bounding boxes from ByteTrack.

We also refer to the reduction performed by Petr Kouba as Kouba-Reduction, see its specification in [1] chapter 4.2.2. We unfortunately do not have the access to the original script and so we can not apply it on the output of HRNet, but we at least have the already calculated output of the Kouba-Reduction applied to OpenPose.

4.4.1 Pair Matching

While we were trying to observe some pattern that would define a dancing pair, we came up with two major ideas on how to identify a dancing pair. The first one is to match the skeletons according to the distance of their hands. From what we observed a dancing pair very often holds their hands in some way. However, they could also hold one hand of each other while stretching the other, therefore we calculate the hand distance of each two skeletons as the minimal distance of each combination of their hands (left-left, right-right, left-right, right-left). It could happen that the pair real pair does not hold their hand whatsoever, in this case we hope that the average distance of their hand will still be lower than compared to others, since they dance close to each other.

The second major idea is to connect a pair based on their bounding box size as we expect the dancing pair to be in the same distance from the camera and thus have a similar size of the bounding box, even tho it could benefit pairing men together and women together, because of their average size in population.

We now multiply the hand distance with the second power of bounding box size difference as it turned out to be the more important factor. To make sure we do not benefit pairs with more common frames, we calculate their score as the average over all common frames. We then apply again the same greedy procedure as used in tracking and match the pairs continuously from the ones with the highest score.

4.4.2 Pair selection

After we successfully paired all the detected skeletons, we now need to select a dancing pair. To do so we tried to utilize various parameters with the trial and error method. In the end we use four parameters, keypoint movement, bounding box movement, bounding box size and the number of common frames. The first three are supposed to reduce the chance of selecting a pair in the audience, the last one to just maximize the number of the data we can work with. We always calculate these parameters for both skeletons in the pair together.

Keypoint movement is calculated as the sum of the distances traveled by all keypoints between 5 consecutive frames. The reason why we skip 4 frames is to benefit those who preserve a continuous movement instead of those just vibrating on the same spot. The downside of this that a poorly detected skeleton, who are vibrating on the same spot, can still achieve quiet high score.

Bounding box movement is just the same as keypoint movement, but we count only with one keypoint, being the middle of the bounding box. This is more robust method for calculating the movement of the pair, but we still skip 4 frames as the bounding box also vibrates on the same spot.

Bounding box size is the sum of the bounding box width and high. We then calculate the average over all frames. The idea behind this is to benefit larger bounding boxes as they are more likely to be a dancing pair, since they are closer to the camera and we expect the camera to focus on the dancers.

The number of common frames is the number of frames, where both of the pair members appear together. The reason for this parameter is to eliminate pairs who could achieve amazing score, but would drastically lower the footage to work with. We count the common frames instead of the total number of frames, because we expect the action recognizer to give better results when evaluating the dancers as a pair together instead of just each dancer separately.

When we finally calculate all these parameters, we normalize across all the dancing pairs, except the number of common frames, we normalize this one using the total number of frames. Then we just multiply these values together and select the pair achieving the highest score. The greatest downside of this method is that we put a lot of emphasis on the movement of the pair, which is relative to the camera. If the camera moves to follow a real dancing pair, the dancing pair is actually not moving much, whereas the audience moves a lot. Therefore we expect this method to work well on the videos with static camera and we hope that if the camera moves, the dancers move enough and are big enough to still outscore the audience.

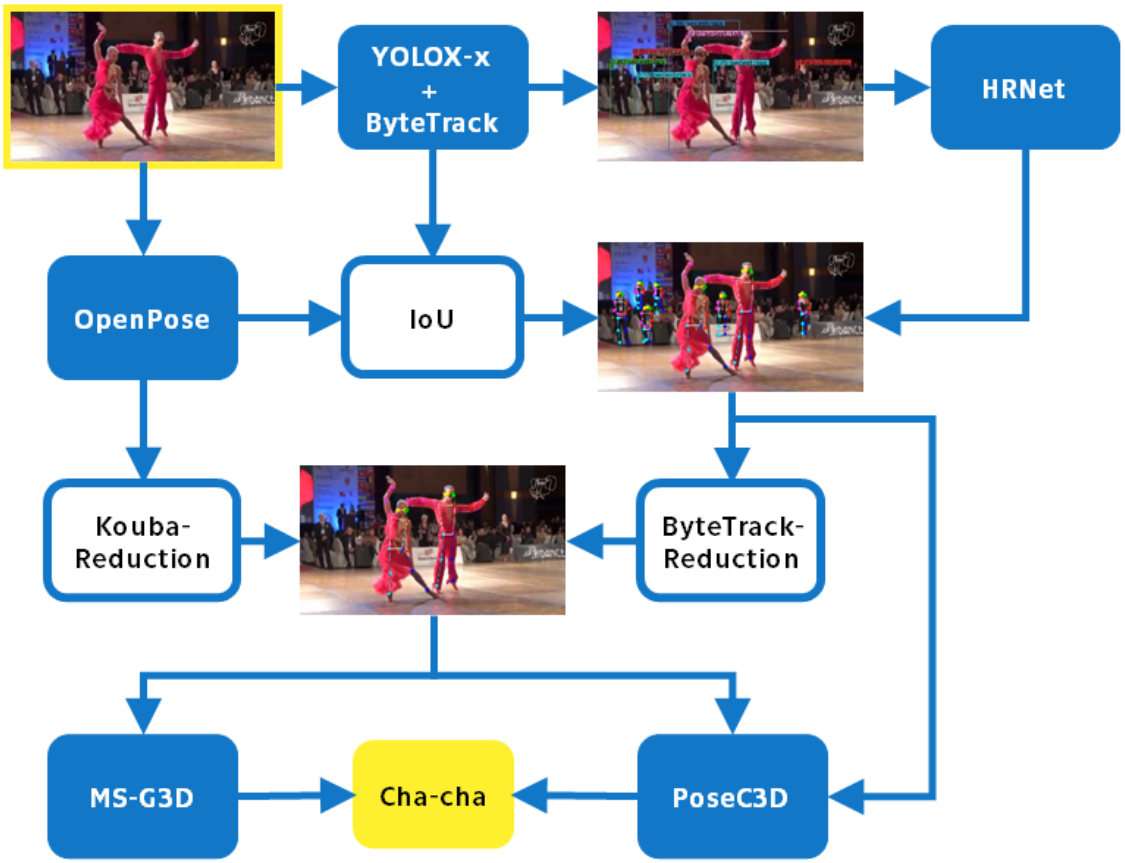

Figure 4.2. Diagram of the examined pipelines.

Figure 4.3. Heatmap visualization. Picture with a skeleton pair on the left, keypoints heatmap in the middle and limbs heatmap on the right.

Chapter 5
Errors

As the methods observed are far from perfect, in this chapter we take a look the most common or interesting errors, which occurred during our testing.

Error type:	Tracking error	Pairing error	Pair selection error
Occurence	46.7%	20.0%	6.7%

Table 5.1. Statistical summary of some the most common errors based on a sample of 30 videos.

5.1 Tracking errors

Most of the tracking errors result from heavy overlapping of the dancers or dancers leaving the scene and therefore being later redetected as a different person. The Figure 5.1 depicts two people being tracked as one person as the lady with big skirt is difficult to be recognized as a human for the object detector.

Figure 5.1. Example of a tracking error caused by overlapping and poorly detected dancers.

5.2 Pair reduction errors

There are many parts that can fail during the pair reduction procedure. It also already relies on the tracking results and as depict in Figure 5.2, the combination of errors results in wrongly selected dancing pair. The reason why this pair was chosen as the best dancing pair is because it is the only pair that appears on many frames as the real dancing pairs just mostly pass by and are always redetected as a new dancing pair and thus have very low percentage of frames they appear on. The camera also moves a little and therefore the audience get some movement score.

As already mentioned, dealing with the moving camera is the greatest challenge and as we can see in the Figure 5.3 the camera slowly transfers from the left frame to the right one, following the dancing pair in the center and thus giving it almost no movement score, unlike the crowd behind which moves quiet a lot, relatively to the camera.

The Figure 5.4 depicts the error of pairing. In the left picture the guys on left are mostly closer to each other than their actual dance partners and are also more similar

in size. In the right picture the dance partners of the selected dancers are completely covered and therefore not detected whatsoever. Therefore it is an example of a pair which was likely paired together in the later stage as the remaining dancers, but then achieved a really good score while selecting the best dancing pair, since that does not count with the pair distance anymore.

Figure 5.2. Example of a pair selection error caused by poor tracking.

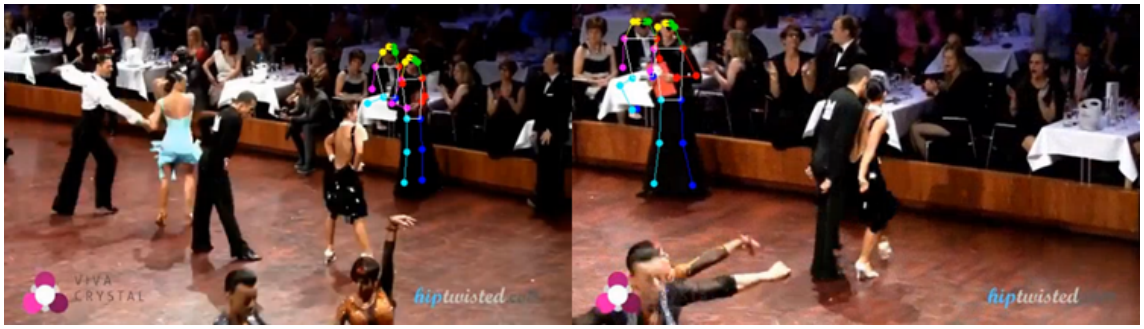

Figure 5.3. Example of a pair selection error caused by a moving camera.

Figure 5.4. Example of pairing errors.

5.3 HRNet pose estimation errors

And lastly we show some of the errors during the pose estimation procedure in the Figure 5.5. The first picture shows a lower body part of one dancer being connected to the upper body part of the other one, which happened for both of the skeletons and are therefore doubled. The second picture shows twisted limbs. The third one wrongly connected keypoints, which is not a real error for PoseC3D. And the last one creating a completely combined skeleton out of two dancers.

Figure 5.5. HRNet pose estimation errors. 1) Doubled skeleton into a single person 2) Switched body parts 3) Wrong keypoint connections 4) One skeleton out of two people

Chapter 6
Experiments

Now when we finally got acquainted with the observed pipelines for skeleton based genre recognition and their possible errors, we can run some tests and see the comparison of all the pipelines and then compare it with the method used in the thesis [1]. However, there is still a lot of factors which can influence the performance the pipelines. And to find the best configuration, we need to test them. In these experiments we will inspect the impact of learning rate and fine tuning various checkpoints retrieved from different datasets. We test all the different configurations on Dance Tutorials Dataset as well as on 10-Let's Dance dataset. All experiments on the 10-Let's Dance dataset were given 300 epochs and all the Dance Tutorials Dataset experiments 1000 epochs. We also provide a preview of the most important sections of a configuration file used to instantiate PoseC3D with different settings 6.1.

6.1 Testing environment

For reproduction purposes of our experiments on PoseC3D, we run them all with the random seed for numpy and pytorch set to 0 and deterministic options for CUDNN backend as described in the documentation[1]. For a better comparison with [1] we originaly wanted to run all the tests on 2 GPUS, unfortunately during the experimenting we discovered a bug, which resulted into running all the experiments on a single GPU and since we already spent a lot of time on these experiments we proceeded to finish the experiments with just a single GPU. Considering the MS-G3D settings, we follow the best resulting ones obtained in the thesis of Petr Kouba [1].

6.1.1 Hyperparameters

We kept all the hyperparameters except learning rate the way they were set by default by the authors of PoseC3D. That means we use a SGD optimizer with a weight decay of 0.0003 and batch size of 16. We only needed to adjust the learning rate as discussed in 6.2

6.1.2 Metrics

There are various methods to measure classification accuracy. The most popular one is probably the Top-k accuracy. The Top-k accuracy will be represented as a percentage of the correctly classified classes, where correctly means that the correct label is within the first k predicted labels. In our experiments we will use the Top-1 accuracy only, despite the Top-2 accuracy or even other Top-k accuracies holding a valuable information too, as it would be rather overwhelming to follow the tables.

[1] `https://mmaction2.readthedocs.io/en/latest/getting_started.html`

```
...
dataset_type = 'PoseDataset'
ann_file_train = 'data/posec3d/gym_train.pkl'
ann_file_val = 'data/posec3d/gym_val.pkl'
left_kp = [1, 3, 5, 7, 9, 11, 13, 15]
right_kp = [2, 4, 6, 8, 10, 12, 14, 16]
train_pipeline = [
    dict(type='UniformSampleFrames', clip_len=48),
    dict(type='PoseDecode'),
    dict(type='PoseCompact', hw_ratio=1., allow_imgpad=True),
    dict(type='Resize', scale=(-1, 64)),
    dict(type='RandomResizedCrop', area_range=(0.56, 1.0)),
    dict(type='Resize', scale=(56, 56), keep_ratio=False),
    dict(
       type='Flip', flip_ratio=0.5,left_kp=left_kp, right_kp=right_kp),
    dict(
        type='GeneratePoseTarget',
        sigma=0.6,
        use_score=True,
        with_kp=True,
        with_limb=False),
    dict(type='FormatShape', input_format='NCTHW'),
    dict(type='Collect', keys=['imgs', 'label'], meta_keys=[]),
    dict(type='ToTensor', keys=['imgs', 'label'])
data = dict(
    videos_per_gpu=16, # Batch size of each single GPU
    workers_per_gpu=2, # Workers to pre-fetch data for each single GPU
    ...
# optimizer
optimizer = dict(
    type='SGD', lr=0.2, momentum=0.9,
    weight_decay=0.0003)  # this lr is used for 8 gpus
optimizer_config = dict(grad_clip=dict(max_norm=40, norm_type=2))
# learning policy
lr_config = dict(policy='CosineAnnealing', by_epoch=False, min_lr=0)
total_epochs = 240 # Total epochs to train the model
checkpoint_config = dict(interval=10)
workflow = [('train', 10)]
evaluation = dict(
    interval=10,
    metrics=['top_k_accuracy', 'mean_class_accuracy'],
    topk=(1, 5))
...
dist_params = dict(backend='nccl')
log_level = 'INFO' # The level of logging
work_dir = './work_dirs/posec3d/slowonly_r50_u48_240e_gym_keypoint'
load_from = None # load models as a pre-trained model from a given path.
resume_from = None # Resume checkpoints from a given path
```

Figure 6.1. An example of a configuration file for MMAction2 [12].

6.2 Learning rate

The first part of our testing aims to find the optimal or more likely suboptimal learning rate. The authors already suggest one for various different setups of the network, but for 8 GPUS. The suggested learning rates may vary a bit, but for our purposes the most important ones were 0.1 and 0.2. Since we use only one GPU, by applying the linear scaling rule we get a suggested learning rates around 0.01 and 0.03. For the fine-tuning purposes we can expect it to be even lower. However these assumptions may be also very wrong and therefore we decided to run most of our experiments for the learning rates of 0.05, 0.01, 0.005, 0.003 and 0.001 to cover a wider interval of possibly suboptimal learning rates

As one can see on the Table 6.1 and the Table 6.3, which were both evaluated on Dance Tutorials Dataset, it seems that the suboptimal learning rate is somewhere around 0.005, since it seems to be the peak of the neighboring learning rates and performs significantly better. However considering the 10-Let's Dance dataset, the Table 6.4 suggests, that even higher learning rate could potentially achieve even better results.

6.3 Datasets split

The second part focuses on different ways to split the datasets. That concerns mostly the Dance Tutorials Dataset. A common technique is to split the dataset into a training part and a validation part at random, while preserving a specific ratio. However as already discussed in 2.1.1 splitting a dataset with varying length of the data samples such as Dance Tutorials Dataset at random could make the learning overfit towards specific genres. In the thesis [1] this problem is mentioned, yet not tested.

To see if the manual Dance Tutorials Dataset split avoids it better, we compare both dataset splits, manual performed for the purposes of [1] as well as random one. We also compare two different approaches for the random split. Both random split approaches split the dataset into 80% for the training part and 20% for the validation part. The first approach splits the dataset by the number of videos per genre in that ratio. The second approach splits the dataset by the length of the total footage per genre in that ratio and cuts the total footage of training dataset for each genre to the lowest one to make the training portion of the footage for all the genres equally long.

Even tho it may seem that the random split approach brings better results according to the Table 6.1 as split by count performs better on its own validation dataset, when we tried to test all three approaches on the 10-Let's Dance dataset as shown in the Table 6.2, we discovered that the model trained on manually split Dance Tutorials Dataset achieves much better reults. Therefore it seems that both of the random split methods tend to overfit more and the consideration mentioned in [1] was right.

Base learning rate:	0.001	0.003	0.005	0.01	0.05
Split by count	55.6%	63.9%	**69.4%**	66.7%	61.1%
Split by footage	53.7%	57.4%	**59.3%**	55.6%	42.6%
Manual split	57.1%	60.0%	**62.9%**	51.4%	40.0%

Table 6.1. Comparison of different approaches to split the Dance Tutorials Dataset, evaluated by the Faster R-CNN + HRNet + PoseC3D fine-tuned on FineGym.

Split method:	Split by count	Split by footage	Manual split
Top-1 accuracy	17.0%	14.0%	**21.6%**

Table 6.2. Comparison of models trained on differently split Dance Tutorials Dataset on the 10-Let's Dance dataset.

6.4 Fine-tuning

The third part compares the PoseC3D part of the model instantiated with three checkpoints, which were trained on three different datasets - Kinetics-400 2.2.2, FineGym 2.2.3 and NTU-60 2.2.4, to see if it has any impact on how well or fast can the model learn. We also decided to include a comparison of the fine-tuning methods with training from scratch.

As shown in the Table 6.3 and the Table 6.4, all three methods preformed quite well, but the one pre-trained on Kinetics-400 2.2.2 is the best and just slightly behind on the 10-Let' Dance dataset and actually equal on the Dance Tutorials Dataset is the model pre-trained on FineGym 2.2.3. NTU-60 2.2.4 fell a bit behind on the 10-Let's Dance dataset and learning from scratch is unsurprisingly last, yet it managed to keep up a bit and with higher learning rate, or given more time, could potentially catch up. We also tried to test all models on Dance Tutorials Dataset to see if one does not tend to overfit more than the others and from what we can see in the Table 6.5, all models kept their positions and thus the model pre-trained on Kinetics-400 2.2.2 seems to be our favorite.

Base learning rate:	0.001	0.003	0.005	0.01	0.05
Kinetics-400	48.5%	57.1%	60.0%	**62.9%**	57.1%
FineGym	57.1%	60.0%	**62.9%**	51.4%	40.0%
NTU-60	54.3%	**60.0%**	57.1%	57.1%	54.3%
Scratch	20.0%	31.4%	**45.7%**	37.1%	45.7%

Table 6.3. Comparison of different fine-tuning approaches on the manually split Dance Tutorials Dataset evaluated by Faster R-CNN + HRNet + PoseC3D.

Base learning rate:	0.001	0.003	0.005	0.01	0.05
Kinetics-400	65.5%	73.1%	74.9%	76.0%	**76.6%**
FineGym	70.8%	**73.1%**	72.5%	67.8%	57.3%
NTU-60	66.7%	63.7%	**68.4%**	66.7%	64.9%
Scratch	35.7%	39.8%	48.5%	51.5%	**57.9%**

Table 6.4. Comparison of different fine-tuning approaches on the 10-Let's Dance dataset evaluated by Faster R-CNN + HRNet + PoseC3D.

Fine-tune method:	Kinetics-400	FineGym	NTU-60	Scratch
Top-1 accuracy	**31.5%**	30.3%	29.6%	26.5%

Table 6.5. Comparison of models learned on the 10-Let's Dance dataset by different fine-tuning approaches on the manually split Dance Tutorials Dataset.

6.5 More errors

But before we get to the pipelines comparison, at this point we realized we did not split the Dance Tutorials Dataset videos by 300 frames, which drastically lowered the results obtained. However, it should not change much about the previous observations and therefore we stick with those results. Since we also already wasted a lot of time and computational power, we still provide some more results obtained on the Dance Tutorials Dataset before splitting it by 300 frames. In the Table 6.6 you can see the pipelines evaluated on the Dance Tutorials Dataset without splitting by 300 frames.

Base learning rate:	0.003	0.005	0.01	0.05
HRNet + ByteTrack-Reduction	61.1%	61.1%	**69.4%**	61.1%
HRNet + No-Reduction	61.1%	**66.7%**	66.7%	58.3%
OpenPose + ByteTrack-Reduction	**58.3%**	52.8%	52.8%	47.2%
OpenPose + No-Reduction	61.1%	**63.9%**	58.3%	55.6%

Table 6.6. Comparison of different pipelines fed into PoseC3D on manually split Dance Tutorials Dataset.

This resulted also into not splitting the videos before transforming them into MS-G3D input format, which we though should perform worse as the ByteTrack-Reduction is performed on the whole video and therefore, there is more segments with missing dancing pairs. After we rerun the tests with applying the ByteTrack-Reduction after the 300 frame split, we compared the results depict in the Table 6.7 and observed that there is no real difference.

Base learning rate:	0.003	0.005	0.01	0.05
HRNet (reduction before split)	52.0%	48.0%	47.1%	**52.9%**
HRNet (reduction after split)	50.0%	**52.8%**	47.2%	47.2%
OpenPose (reduction before split)	44.1%	47.1%	**49.0%**	40.2%
OpenPose (reduction after split)	44.4%	38.9%	**47.2%**	33.3%

Table 6.7. Comparison of applying the ByteTrack-Reduction before and after dividing the videos into 300 frame segments, evaluated on Dance Tutorials Dataset with MS-G3D..

6.6 Pipelines

Now we finally get to the most important part of the tests, combining almost all components of the pipeline presented in the work [14] and [1]. For a clear comparison we rerun even the pipeline of Petr Kouba again, using the settings that achieves the best results in his work [1]. It achieves slightly worse results than presented in his work. But still decent enough so we do not consider it as an error.

In the Table 6.8 and the Table 6.9 we can see that the most dominant methods on Dance Tutorials Dataset are those using PoseC3D, specifically with no reduction into a single dancing pair. The ByteTrack-Reduction performs slightly better with PoseC3D than the Kouba-Reduction but worse with MS-G3D. We can also observe that HRNet is performing slightly better than OpenPose with both action recognition networks.

Now if we take a look at the 10-Let's Dance dataset compared in the Table 6.10 and Table 6.11, we can observe again dominance of the PoseC3D network and again with

Base learning rate:	0.003	0.005	0.01	0.05
HRNet + ByteTrack-Reduction	68.6%	68.6%	68.7%	**71.6%**
HRNet + No-Reduction	72.6%	73.5%	**74.5%**	72.6%
OpenPose + ByteTrack-Reduction	62.8%	**67.7%**	66.7%	63.7%
OpenPose + No-Reduction	70.6%	**72.6%**	72.6%	72.6%
OpenPose + Kouba-Reduction	**68.8%**	66.4%	63.2%	64.8%

Table 6.8. Comparison of different pipelines fed into PoseC3D on manually split Dance Tutorials Dataset divided into 300 frames.

Base learning rate:	0.003	0.005	0.01	0.05
HRNet + ByteTrack-Reduction	50.0%	**52.8%**	47.2%	47.2%
OpenPose + ByteTrack-Reduction	44.4%	38.9%	**47.2%**	33.3%
OpenPose + Kouba-Reduction	**60.0%**	56.8%	60.0%	58.4%

Table 6.9. Comparison of different pipelines fed into MS-G3D on manually split Dance Tutorials Dataset divided into 300 frames.

Base learning rate:	0.005	0.01	0.05	0.1
HRNet + ByteTrack-Reduction	**55.4%**	50.9%	50.9%	41.8%
HRNet + No-Reduction	**77.4%**	74.0%	74.6%	62.2%
OpenPose + ByteTrack-Reduction	40.7%	40.7%	**44.1%**	36.2%
OpenPose + No-Reduction	69.5%	**76.8%**	75.1%	57.6%
OpenPose + Kouba-Reduction	**57.1%**	51.7%	56.6%	42.3%

Table 6.10. Comparison of different pipelines fed into PoseC3D on the 10-Let's Dance dataset.

Base learning rate:	0.005	0.01	0.05	0.1
HRNet + ByteTrack-Reduction	**54.4%**	54.4%	50.0%	25.4%
OpenPose + ByteTrack-Reduction	**44.6%**	41.1%	36.6%	26.8%
OpenPose + Kouba-Reduction	50.0%	**51.1%**	42.9%	23.1%

Table 6.11. Comparison of different pipelines fed into MS-G3D on the 10-Let's Dance dataset.

no pair reduction. However, in this example we see that the pair reduction methods do not perform that much better with PoseC3D, than with MS-G3D. HRNet again slightly but still outperforms OpenPose.

6.7 Comparison to the Kouba results

As we saw on the previous results. The best model is by far HRNet with No-Reduction fed into PoseC3D and thus it is out best candidate to compare with the results obtained by Petr Kouba [1]. We compare the models based just on the visual information only and without applying sliding-window [1]. As we can see in the Table 6.12, the PoseC3D model strongly outperforms the one introduced in Kouba's work as it shows increase

by about 10% on the Dance Tutorials Dataset and about 22% on the 10-Let's Dance dataset.

Considering their run time speed, since OpenPose with PoseC3D achieves just slightly worse results, we could exchange it in case it performs way faster than the proposed HR-Net. Therefore it all comes down to MS-G3D and PoseC3D. From our experiments we noticed, that the PoseC3D network is quiet slower than the MS-G3D with low amount of skeletons, however it scales way better with large amount of skeletons, because of its CNN base, which is not being extended.

Dataset:	Dance Tutorials Dataset	10-Let's Dance Dataset
OpenPose + MS-G3D [1]	64.8%	55.5%
HRNet + PoseC3D	**74.5%**	**77.4%**

Table 6.12. Comparison of the best model from our pipeline with the OpenPose + MS-G3D model from the thesis of Petr Kouba [1].

Chapter 7
Conclusion

In this thesis we follow up on the recent work [1], which proposes a GCN-based action recognition method with OpenPose estimator and implements its own tracking method to solve the difficult problem of the dance genre recognition. We propose a new model using HRNet with no tracking and PoseC3D. This model according to the results shown in the Table 6.12 performs far better on the Dance Tutorials Dataset as well as 10-Let's Dance dataset and thus has a great potential. It achieves results beyond our expectations as the Top-1 accuracy was improved by roughtly 10% on the Dance Tutorials Dataset and by roughly 22% on the 10-Let's Dance dataset. This number may not be exact as both implementations rely on various settings.

This new model is a proposal for skeleton based action recognition by the work [14]. It substitutes OpenPose with external tracking method for HRNet to improve the pose estimation of dancers. It also replaces the GCN-based action recognition with more robust 3D-CNN-based network, PoseC3D. Further we tried to optimize very few of its parameters, but kept most of the model untouched and as presented by [14].

We implemented a new tracking method for reducing the crowd of people on a video into a single dancing pair. This method suffers mostly from a moving camera, which could possibly be solved by applying optical flow methods to calculate the movement of the camera. However, it also turned out that the best model just does not use tracking of a dancing pair at all, which is quiet surprising and would deserve further investigation.

We also tried to compare different approaches of splitting the datasets to see if we can get more of it. That was unfortunately without success, but we at least confirmed our expectations. The datasets seem to however be a bit of a problem as none of them seems to be big and complex enough to be a good candidate for pre-training a model, which could withstand real-world data.

As we compared only the base of our models, there is still a lot of space for further improvements of this model. We would recommend to also apply the sliding-window method as proposed in [1]. In this thesis we also did not try the combination with any audio-based dance genre recognition method. An application of such audio-based method would also definitely further improve the model.

References

[1] Petr Kouba. *Recognition of Dance Genres from Video. Czech Technical University, FEE*. 2021.
`https://dspace.cvut.cz/bitstream/handle/10467/97076/F3-DP-2021-Kouba-Petr-PetrKoubaThesisFinal.pdf`.

[2] Tomáš Pavlín. *Dance Recognition from Audio Recordings. Charles University, MFF*. 2020.
`https://dspace.cuni.cz/handle/20.500.11956/116600`.

[3] OpenMMLab contributors. *OpenMMLab*.
`https://github.com/open-mmlab`.

[4] Daniel Castro, Steven Hickson, Patsorn Sangkloy, Bhavishya Mittal, Sean Dai, James Hays, and Irfan A. Essa. Let's Dance: Learning From Online Dance Videos. *CoRR*. 2018, abs/1801.07388

[5] (Facebook AI). *COCO (Microsoft Common Objects in Context)*.
`https://paperswithcode.com/dataset/coco`.

[6] (Facebook AI). *Kinetics (Kinetics Human Action Video Dataset)*.
`https://paperswithcode.com/dataset/kinetics`.

[7] (Facebook AI). *FineGym*.
`https://paperswithcode.com/dataset/finegym`.

[8] (Facebook AI). *NTU RGB+D*.
`https://paperswithcode.com/dataset/ntu-rgb-d`.

[9] MMDetection Contributors. *OpenMMLab Detection Toolbox and Benchmark*. 2018.
`https://github.com/open-mmlab/mmdetection`.

[10] MMPose Contributors. *OpenMMLab Pose Estimation Toolbox and Benchmark*. 2020.
`https://github.com/open-mmlab/mmpose`.

[11] MMTracking Contributors. *OpenMMLab Video Perception Toolbox and Benchmark*. 2021.
`https://github.com/open-mmlab/mmtracking`.

[12] MMAction2 Contributors. *OpenMMLab's Next Generation Video Understanding Toolbox and Benchmark*. 2020.
`https://github.com/open-mmlab/mmaction2`.

[13] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.*. 2017, 39 (6), 1137–1149. DOI 10.1109/TPAMI.2016.2577031.

[14] Haodong Duan, Yue Zhao, Kai Chen, Dian Shao, Dahua Lin, and Bo Dai. Revisiting Skeleton-based Action Recognition. *CoRR*. 2021, abs/2104.13586

[15] (Facebook AI). *Real-Time Object Detection*. https://paperswithcode.com/task/real-time-object-detection.

[16] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *CoRR*. 2015, abs/1506.02640

[17] (Facebook AI). *Pose Estimation on MPII Human Pose*. https://paperswithcode.com/sota/pose-estimation-on-mpii-human-pose.

[18] (Facebook AI). *Multi-Person Pose Estimation*. https://paperswithcode.com/task/multi-person-pose-estimation.

[19] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep High-Resolution Representation Learning for Human Pose Estimation. *CoRR*. 2019, abs/1902.09212

[20] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *CoRR*. 2018, abs/1812.08008

[21] Bin Xiao, Haiping Wu, and Yichen Wei. Simple Baselines for Human Pose Estimation and Tracking. *CoRR*. 2018, abs/1804.06208

[22] (Facebook AI). *Pose Tracking on PoseTrack2017*. https://paperswithcode.com/sota/pose-tracking-on-posetrack2017.

[23] (MOT Challenge). *MOT Challenge*. https://motchallenge.net/.

[24] (Facebook AI). *Multi-Object Tracking on MOT17*. https://paperswithcode.com/sota/multi-object-tracking-on-mot17.

[25] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. BoT-SORT: Robust Associations Multi-Pedestrian Tracking. *CoRR*. 2022, abs/2206.14651 DOI 10.48550/arXiv.2206.14651.

[26] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. *CoRR*. 2021, abs/2110.06864

[27] (Facebook AI). *DanceTrack* . https://paperswithcode.com/dataset/dancetrack.

[28] (Facebook AI). *Skeleton Based Action Recognition on Kinetics-Skeleton dataset*. https://paperswithcode.com/sota/skeleton-based-action-recognition-on-kinetics.

[29] (Facebook AI). *Revisiting Skeleton-based Action Recognition*. hhttps://paperswithcode.com/paper/revisiting-skeleton-based-action-recognition.

[30] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. *YOLOX: Exceeding YOLO Series in 2021*. 2022. https://arxiv.org/pdf/2107.08430.pdf.

[31] (Facebook AI). *YOLOX: Exceeding YOLO Series in 2021*. https://paperswithcode.com/paper/yolox-exceeding-yolo-series-in-2021.