Master Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Recognition of Dance Genres from Video

Petr Kouba, MSc.

Supervisor: prof. Ing. Jiří Matas, PhD.
August 2021

Acknowledgements

Firstly, I would like to thank prof. Matas for many inspiring suggestions and for the overall supervision of this thesis.

Secondly, I would like to thank Tomáš Pavlín for providing the audio-based dance classifier and Štěpán Křivánek for providing me with useful feedback regarding this text.

Last but not least, I thank Anna for her patience and support.

The access to the computational infrastructure of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 “Research Center for Informatics” is also gratefully acknowledged.

Declaration

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, 13.8.2021

Signature: ....

Petr Kouba

Abstract

We propose a method for recognition of ballroom dance genres from video. The method is based on processing of videos by a human pose estimation framework OpenPose and a subsequent classification of the sequences of the estimated poses using a Graph Convolutional Neural Network called MS-G3D. For the purposes of the method development we collected three suitable video datasets which we make publicly available. On a set of videos containing a clearly visible dance couple, undisturbed by presence of other people, the method achieves Top-1 accuracy 72.2 %. The classification accuracy improves to 83.3 % when the method is combined with an existing audio-based dance classifier (which on its own reaches an accuracy of 66.7 % on the dataset).

Keywords: Dance, Recognition, OpenPose, GCN, Multi-Modal, Classification

Supervisor: prof. Ing. Jiří Matas, PhD.
Karlovo náměstí 13,
121 35 Praha 2

Abstrakt

Předkládáme metodu rozpoznávání společenských tanců z videa. Metoda je založena na využití nástroje OpenPose, který odhaduje postoj snímaných tanečníků v jednotlivých snímcích videa. Následně klasifikujeme posloupnosti odhadnutých postojů za pomoci grafové konvoluční neuronové sítě MS-G3D. Za účelem vývoje této metody jsme sestavili tři vhodné sady videí, které nabízíme veřejně k dispozici. Na sadě videí, zobrazujících jasně jeden taneční pár a nenarušených přitomností jiných osob, dosahuje navrhovaná metoda přesnosti klasifikace 72.2% (Top-1). Přesnost klasifikace se zvyšuje na 83.3 % (Top-1) pokud metodu spojíme s existující metodou klasifikující tanec na základě audia (sama o sobě dosahuje tato audio-klasifikace na zmiňované datové sadě přesnosti 66.7 % (Top-1)).

Klíčová slova: Rozpoznávání, Tanec, OpenPose, GCN, Multi-modální, Klasifikace

Překlad názvu: Rozpoznávání tance z videa

Contents

Figures

Tables

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Kouba Petr**	Personal ID number:	**492585**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Specialisation:	**Computer Vision and Image Processing**		

II. Master’s thesis details

Master’s thesis title in English:

Recognition of Dance Genres from Video

Master’s thesis title in Czech:

Rozpoznávání tance z videa

Guidelines:

1. Propose an approach for ballroom dance recognition that is based on visual iformation only. The method will be tested on youtube videos with one or more dancing pairs.
2. Collect a suitable collection 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].
3. As a core of the approach, apply the OpenPose algorithm for pose estimation. Solve the problem of tracking of individual dancers and their clustering into 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.
6. Optionally, combine the result with the audio based method of T. Pavlín and discuss the performance and failures of the joint system.

Bibliography / sources:

[1] Tomáš Pavlín: Dance Recognition from Audio Recordings (diplomová práce, 2020)
[2] 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 master’s thesis supervisor:

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

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

Date of master’s thesis assignment: **07.01.2021** Deadline for master's thesis submission: **13.08.2021**

Assignment valid until: **30.09.2022**

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 master’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 master’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt Student’s signature

Chapter 1

Introduction

One of the interesting aspects of dance recognition is that it can be done based on two very distinct modalities. In a scene with dance being perfomed, music can usually be heard as well. Both the movements of the dancers perceived visually and the music perceived auditorily provide the spectator with information about the dance.

A recent work [15] introduces an audio-based approach to the classification of ballroom dances belonging to the so-called International Style[1], see Table 1.1. For simplicity, we call the dances listed in Table 1.1 'The 10 Dances'.

In this work we propose a method for classification of video recordings of The 10 Dances based on visual information only. Combining the proposed visual-based classifier with the audio-based classifier [15] allows for studying how the two modalities complement each other and how beneficial it is to combine both of the classifiers.

The proposed visual-based method relies on deep learning. A crucial element in development of a deep learning based classifier is the availability of sufficient amount of data. To this end, we review the relevant existing datasets in Section 2.1 and introduce three new datasets specifically for the purpose of classification of The 10 Dances in Section 2.2.

The multi-modal character of dance is of interest even in the data collection phase, as one of the modalities can be used in annotation of the datasets for the other modality. More details on this aspect is provided in 2.2.3.

The task of visual-based dance recognition can be viewed as a subtask of the area of Human Action Recognition. This broader research field has a great potential for applications in different areas such as visual surveillance, entertainment and abnormal activity detection [1]. Driven by the active research, different approaches towards Human Action Recognition were proposed [1].

In this work we follow the direction of the so-called Skeleton-Based action recognition, which we review in Section 3.2.2. This approach builds upon Human Pose Estimation - a research field reviewed in Section 3.3. Our

[1]Dance styles as defined by the 2015 competition rules of the World Dance Council: `https://web.archive.org/web/20150910165950/http://www.dancewdc.org/downloads/WDC%20Competition%20Rules%20current%20June%202015.pdf`, Accessed 8/12/2021

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

Table 1.1: Dance styles belonging to 'International Style'. They can be divided into two kinds - latin and standard. In this work we further refer to these dance styles as 'The 10 Dances'.

motivation for the choice of this approach is to find out whether dance can be represented sufficiently well using sequence of estimated human poses.

In Chapter 4 we provide the details of the proposed method. The achieved results are summarized in Chapter 5. Among other things we evaluate the method on the datasets introduced in Section 2.2, we study the influence of duration of the dance on accuracy of the model and we evaluate the benefits of multi-modal classification, performed with the help of the audio-based model [15].

Chapter 2

Data

Even though the task of dance classification from video is rather specific, some datasets relevant to this task are available. However, no dataset fits exactly the task of clasifying The 10 Dances. Moreover, it is beneficial to have diverse datasets, since it enables us to study how the classifier behaves based on the differences between datasets. To this end, we introduce three datasets of our own.

2.1 Existing datasets

We make use of two large-scale datasets: *Youtube-8M* [2] and *Kinetics* [6]. Even though they are not focused on dance, they are still valuable to us as we can use their relevant subsets or utilize them for pretraining. Specifically for the purpose of dance recognition we make use of the dataset *Let's Dance* [13] which covers several types of dance.

2.1.1 Youtube-8M

The original version of Youtube-8M consists of 8 million videos freely available on Youtube. To each video, one or more of the 3800 labels was assigned using the Youtube video annotation system [3]. The labels are based on the supposed main topics of the video which could be human actions but also just some objects. The labeling itself is semi-automatic, because the annotation system relies heavily on the video metadata - such as the title, description, keywords etc. - which are mainly entered by the users .

The authors released the official version of the dataset in the form of precomputed audio-visual features of the videos in 2016, in order to compress the enormous amount of data and make the training more accessible to researchers. There is no official support providing the raw video URLs with its labels, however they can be extracted from the dataset using an existing unofficial tool [44].

In 2018 the dataset was updated. The quality of machine-generated labels was improved and the new dataset to this date consists of 6.1 million videos and 3862 different lables [4]. Note that both of the versions are called the same. We work with the version of 2018 unless stated otherwise.

In 2019, new derived dataset *Youtube-8M Segments* was introduced. This one is special by smaller granularity of the labelling, where individual segments of the videos have their human verified labels - more interesting data at the cost of the size of the dataset. *Youtube-8M Segments* has 1000 different labels and 237 000 labeled video segments, with 5 segments per video on average [4].

Although *Youtube-8M* can be - to some extent - helpful in development of human action classifiers, it is meant to be more general and due to the missing temporal localization of the labels in the videos it is not entirely tailored to the needs of the human action recognition research and some more work has to be put in to utilize it in this work. *Youtube-8M Segments* on the other hand, has a good temporal localization of the labels but as none of the labels corresponds to ballroom dance we do not use this dataset at all.

2.1.2 Kinetics

The original version of the *Kinetics* dataset, nowadays referred to as *Kinetics-400*, is based on 306 000 videos [6] [5] and it has 400 different human-annotated classes, each corresponding to a specific human action, e.g.: 'crying', 'brushing hair', 'drinking beer' etc. . Every class has 400-1150 instances - each corresponding to an approximately 10 second long clip, each from a unique video.

Kinetics aspires to have the role of a basis for pretraining of the human action classifiers, similarly to the role the Imagenet [7] dataset plays in the image recognition domain.

Since the 2017 release of *Kinetics-400*, an updated version of the dataset gets released each year.

The *Kinetics-600* [8] has 368 classes from *Kinetics-400* with the remaining 32 classes renamed, split or removed. Furthermore, it adds new classes to have the total of 600 classes with at least 600 instances each.

Kinetics-700 [9] introduces again new classes to reach the number of 700 classes, however it stays with the minimum of 600 instances per class. The most recent version *Kinetics-700-2020* [10] revises the numbers of instances per class, so that each class has at least 700 instances.

Kinetics-Skeleton

The authors of [11] processed the *Kinetics-400* dataset by using OpenPose and created a new representation of the dataset in the form of skeletons each consisting of 18 keypoints. They call this representation *Kinetics-Skeleton* and make it publicly available in order to facilitate the research in the field of Skeleton-Based Human Action Recognition. To the best of our knowledge, no such representation of *Kinetics-600*, *Kinetics-700* nor *Kinetics-700-2020* was made publicly available.

Original dataset classes		**Classes added later**
Ballet	Swing	**Jive**
Flamenco	**Foxtrot**	**Samba**
Latin	**Tango**	**Paso Doble**
Square	**Quickstep**	**Rumba**
Break Dancing	**Waltz**	**Cha-Cha**
		Tap dance

Table 2.1: Classes included in the *Let's Dance* dataset. Classes which also belong to The 10 Dances are written in bold. Note that 'Foxtrot' in *Let's Dance* is equivalent to 'Slowfox' in The 10 Dances.

2.1.3 Let's Dance

Out of the publicly available data collections, we were able to identify only one dataset focused on dance with classes significantly overlapping with The 10 Dances - the so-called *Let's Dance* dataset [13].

The original version of *Let's Dance* has 10 classes with 100 different dance clips per each class, each clip taking 10 seconds [13]. Later the dataset was updated and it currently has 16 classes, unfortunately we could not find any documentation of this update. The classes constituting the dataset are provided in Table 2.1. We can see that 9 of the classes belong to The 10 Dances, see Table 1.1. Out of The 10 Dances, only the Viennese Waltz is not covered by *Let's Dance*.

The dataset is released in three different representations. First representation are the bare sequences of RGB frames which - as opposed to videos - do not contain any audio. Secondly, the dataset was represented by optical flows. The last representation uses skeletons, however they used the pose estimation framework called DensePose [14] which produces skeletons incompatible with our preferred format of pose representation. Since our preferred representation of the dataset is in the form of videos, we will describe a method of obtaining those in Section 2.2.2.

2.2 New datasets

We prepared three datasets precisely covering The 10 Dances. Firstly, we introduce *Dance Tutorials Dataset* consisting of video segments which are the easiest to classify. Secondly, we present the *10-Let's Dance* dataset with more challenging dance recordings of multiple dance couples. Thirdly, we create a dataset called *YT8M Ballroom* using a noisy semi-automatic labeling procedure.

Comparison of the performance of our model using different datasets can tell us something about the effect the data properties have on the classifier. For example, it enables us to address the question whether it pays off to collect more data at the cost of the lower quality of its labeling.

2.2.1 Dance Tutorials Dataset

We introduce *Dance Tutorials Dataset* which we produced by collecting Youtube videos and choosing their relevant segments. We searched for videos on Youtube using the search terms corresponding to the name of the dance style or the name of the dance style followed by 'routine'. We watched the whole candidate videos and we indicated segments of interest. To decide whether we will include the video or its segment in the dataset, we checked whether it meets the following rules:

- The video segment contains only one dancing couple. Reflection of the couple in a mirror is allowed.
- The video segment does not contain other people, like the audience or photographers. In rare cases we accept video segments with one extra person only occurring in a fraction of the recording.
- The video segments contain only dancing. Exclude e.g. frames of the dance couple standing while waiting for the music etc.
- Include maximum of 60 seconds from a single dance performance, to keep the dataset diverse.
- Include maximum of 80 seconds total from a single video (achievable with one dance couple showing more dance performances in a single video)
- For each of The 10 Dances the total footage of all instances has to add up to at least 10 minutes
- Construct the total footage of each class from as many videos as possible to achieve maximum diversity of each class.

In most cases, the source of the complying videos were online dance instructors teaching basic steps but sometimes they presented even advanced dance routines. Often only a minor part of the videos contained the actual dance - it was typically accompanied by a video intro and the tutor standing and explaining the coming dance.

Due to the lack of our own domain expertise we annotate the clips by the dance style according to what the protagonists or the video title claimed.

For each class we keep a separate text file, where each line corresponds to a single instance. The instance is represented by the ID of the Youtube video, followed by numbers separated by spaces. These numbers denote the start and end times of relevant segments in seconds, see example in Figure 2.1.

In Table 2.2, we provide the summary of the class distribution of *Dance Tutorials Dataset*. We make the dataset publicly available in the form of text files illustrated in Figure 2.1 together with a Python script which we prepared and which downloads all the videos, edits them and stores the relevant segments [49].

```
'7tJBYwrZxaw 8 48'
```

(a) : Instance created from a video with Youtube ID '7tJBYwrZxaw' corresponding to the segment starting at 8^{th} second and ending at 48^{th} second.

```
'l1JPLXKWp2U 166 194 210 234'
```

(b) : Instance created from a video with Youtube ID 'l1JPLXKWp2U' corresponding to the 2 segments 166s to 194s and 210s to 234s respectively.

Figure 2.1: Examples of the representations of the *Dance Tutorials Dataset*'s instances in the form of text files.

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 2.2: Classes in *Dance Tutorials Dataset*, total footage of their instances and number of different videos from which the instances were obtained.

2.2.2 10-Let's Dance

The *10-Let's Dance* dataset is a modification of the *Let's Dance* dataset described in Section 2.1.3. This modification is based on the 9 of The 10 Dances covered by *Let's Dance* plus a collection of Viennese Waltz recordings which we created from segments of Youtube videos.

As mentioned in Section 2.1.3, the released form of *Let's Dance* misses the representation in the form of the actual videos or links to the videos. In order to have a dataset which can potentially serve for benchmarking multimodal audio-visual classifiers we are interested in having the actual videos. Luckily, the IDs of the Youtube videos the start times of the relevant segment and the frame number since the segment beggining are encoded in the names of the instances in the RGB frames representation of the dataset. Using a Python script we created, we parsed the instance names and obtained the Youtube IDs, start times and frame numbers of the selected video segments. We downloaded the videos and using OpenCV [45] we read out their frame rate. Knowing the frame rate we converted the text representation of the instances into the same format as described in Figure 2.1a. We make this representation publicly available together with a downloader script and the script for parsing the names of the original *Let's Dance* instances [49].

The typical instance of the *10-Let's Dance* dataset is a recording from a ball or a dance competition where there are several dance couples on the dance floor at the same time with spectators surrounding them. Such a scene resembles a real life scenario in which it would be desirable to have a dance

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 2.3: Classes in *10-Let's Dance*, total footage of their instances and number of instances per class. Some of the instances from *Lets Dance* were not available due to territorial copyright restrictions or due to the video being removed from Youtube.

classifier.

The typical length of the video segments corresponding to the instances is around 10 seconds (since usually there are 301 RGB frames per instance in *Let's Dance*). To keep consistency, the complementing Viennese Waltz clips are thus 10 seconds in length as well. The dataset statistics is provided in 2.3.

2.2.3 YT8M Ballroom

YT8M Ballroom is a new experimental dataset that is one of our novel outputs. In contrast to *Dance Tutorials Dataset* and *10-Let's Dance* datasets which rely on time demanding manual annotation, we developed a semi-automated annotation procedure that allows for noise in the annotation and makes it possible to create much larger data collections. *YT8M Ballroom* has been created using this procedure. We use *YT8M Ballroom* to investigate whether we can profit from such sizeable - yet noisily annotated - dataset.

Preselection of videos

First, we collect the IDs of Youtube videos where we suspect one of the ballroom dances corresponding to our classes might be occuring. To this end, we utilize the *Youtube-8M* dataset described in 2.2.3. It contains 2 relevant classes: 'Ballroom dance' and 'Latin Dance'. There are several videos labelled with both 'Ballroom dance' and 'Latin dance' at the same time. We drop these duplicates by defining the set of our candidate videos as an union of these two classes and we arrive at the set of 4548 preselected video IDs.

Audio-based annotation of videos

In order to both locate a video segment where a dance performance is supposedly occurring and annotate the segment by a label, we utilize the audio-based ballroom dance classifier [15]. This classifier is trained to distinguish The 10 Dances, it ignores any visual information and relies solely on music.

Since our preselected videos come from quite general classes of 'Ballroom dance' and 'Latin dance', there is a chance that a dance outside of The 10 Dances can occur[1]. Unfortunately, the audio-based classifier does not have any reject option which would identify an occurrence of an unknown dance. The authors of [15] are aware of this limitation and as a workaround they propose the analysis of the entropy of the output class probability distribution.

We followed the recommendation and utilized Shannon entropy (2.1) [16] to evaluate the certainty of the audio classification.

$$S(r) = -\sum_{c \in \mathcal{C}} p_r(c) \cdot \log(p_r(c)) \tag{2.1}$$

In (2.1) $p_r(c)$ is the probability distribution over The 10 Dances $\mathcal{C}$ corresponding to the softmax scores returned by the audio-classifier for a given audio segment r. When $p_r(c) = 0$, we replaced the summand by its right-hand limit[2]:

$$\lim_{p_r(c) \to 0^+} p_r(c) \cdot \log(p_r(c)) = 0 \tag{2.2}$$

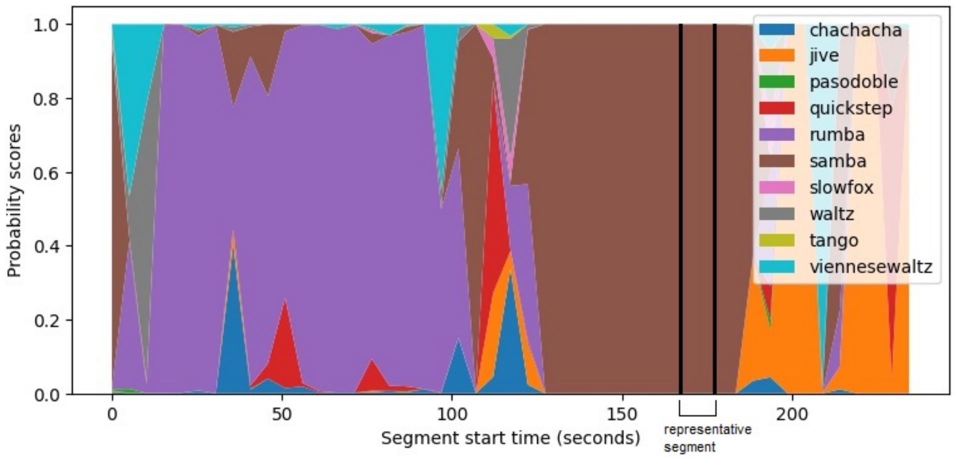
If the entropy for a given audio segment r is high, we interpret it such that the classifier is not confident about its result. We set a threshold on the entropy and we disregard the segments scoring above the threshold as likely misclassified. We observe that this high entropy is typical for segments with no relevant dance music playing at all - due to the missing reject option however still classified as one of The 10 Dances.

We utilize the described uncertainty assessment to identify promising segments of each video. To achieve that, we apply the audio-classifier not to the videos as a whole but instead in sliding window manner only to its fixed length segments. The recommended window length leading to reliable classification is 5.2 seconds [15]. Using this length we slide the window across the recordings with a stride corresponding to the segment length - resulting in non-overlapping segments. For each two consecutive segments which roughly correspond to a 10 second clip, we average the output probability distributions and compute its entropy (2.1). Each of the videos is then represented by the approximately 10 second long segment with the lowest entropy. For illustration of the sliding window audio-classification, see Figure 2.2.

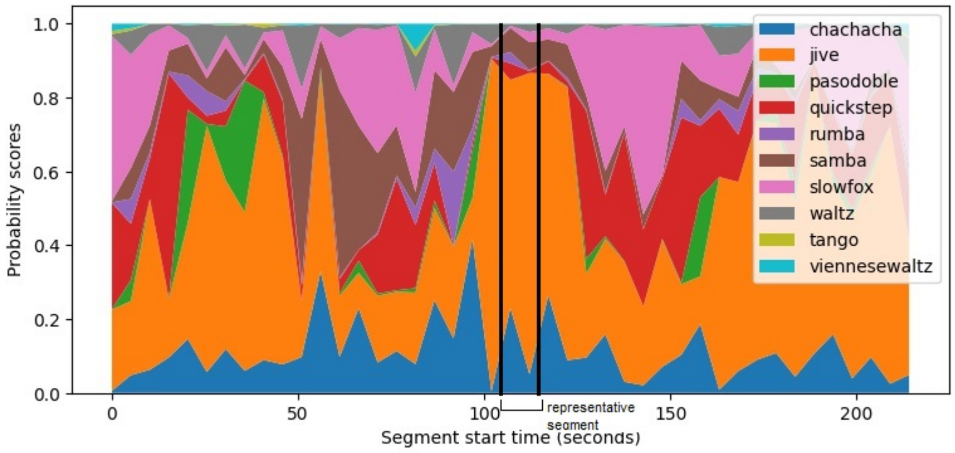
To discard the videos where it is unlikely that a relevant dance occurs, we sort all of the videos by the entropy of its representative segment. We select an arbitrary threshold on how many videos we would like to pass this filter

[1]To name two such dances, Bachata and Salsa occurred among the videos in *Youtube-8M* labeled as 'Latin dance' rather often.

[2]Can be computed using the rule of L'Hospital



(a) : Example of a video with a confidently classified audio (`https://www.youtube.com/watch?v=2Ocpq-tzuYY`). The representative segment with the lowest entropy is denoted in the plot. In this case the audio classfication worked great, because in the video rumba, samba and jive indeed all occur one after another - as the plot suggests.



(b) : Example of a video with a low confidence in audio classification (`https://www.youtube.com/watch?v=z4WCC_cZJWY`). The representative segment with the lowest entropy denoted in the plot. In the video actual latin dance music plays, but it is not clear for which dance it is best suited - the dancers perform some dance improvisation.

Figure 2.2: Example stack plots of audio-classification of 2 different videos from the union of 'Ballroom dance' and 'Latin dance' in *Youtube-8M*. Classification performed by non-overlapping sliding windows of 5.2 second. Credit for the stack plot design is due to [15]

(in our case 3000) and the remaining videos with the highest entropy are dropped.

When creating this dataset, we rely on a very strong assumption that there are people in the video who dance to the correctly classified music. This assumption is partly reasonable, since from the *Youtube-8M* label we know there should be a dance performance contained [3] in the video and this likely happens with a music playing on the background. Quite often however, the assumption fails - e.g. when:

- An incorrectly classified audio passed our entropy threshold
- People in the video do not care about the music and perform an irrelevant dance
- There can be an intro or outro with a dance music correctly classified by the audio-classifier but the actual dance is performed without music (for example the dancers just counting the beat)

In practice, the listed cases are not rare. Especially our entropy bar is often met even by dance recordings outside of The 10 Dances - for example Bachata and Salsa often end up classified as Cha-Cha with high confidence.

The final version of *YT8M Ballroom*

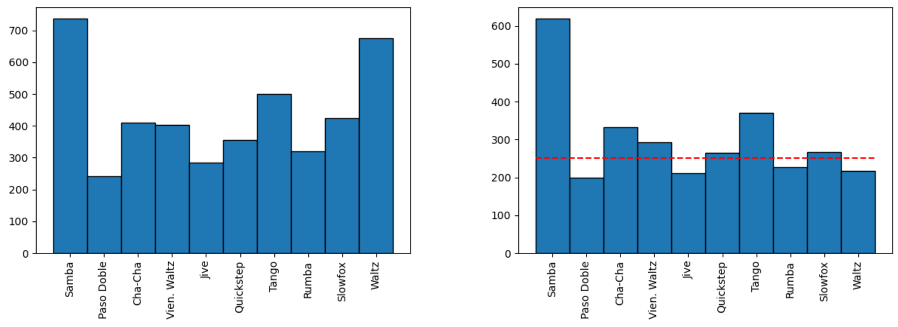
Following the above described procedure, we first preselected 4548 Youtube videos which we tried to download. Due to some videos being removed or due territorial copyright protection we succeeded in obtaining only 4353 out of them.

After performing the entropy analysis of the audio-classification, we kept 3000 video segments with the lowest entropy - each representing different video. See Figure 2.3, for comparison of class distributions of the dataset before and after the removal of high entropy samples.

To have a balanced dataset, we decide to randomly sample from the more populous classes so that no class holds more than 251 instances (see Figure 2.3b). The final dataset statistics can be seen in Table 2.4. We make available[4] the text files with the relevant Youtube IDs and segment times together with the download script as well as the softmax scores for all of the segments of all of the videos.

[3]When inspecting the dataset manually, we found videos which passed the *Youtube-8M dataset* annotation, even though they only contain a static image with music playing on the background. This is probably not intended by the authors of the dataset and it is a result of the annotation system relying on the video metadata. In our case, we can filter these few videos out in the skeletonization step - no skeletons should be detected in those photos (since usually these photos contain just a text with the name of the song etc.)

[4]`https://github.com/KoubaPetr/BallroomDanceDatasets/tree/main/YT8MBallroom`



(a) : Class distribution of all of the 4352 obtained videos.

(b) : Class distribution of the 3000 segments with best entropy. We choose this subset as the basis for our dataset. Since such dataset would be unbalanced, we will cap the number of instances per class in the final dataset (limit corresponding to the dashed line).

Figure 2.3: Class distribution of the *YT8M Ballroom* dataset before and after the drop of high entropy videos.

Genre	**Total footage**	**Number of instances**
Cha-Cha	43:31	251
Slowfox	43:31	251
Jive	36:45	212
Samba	43:31	251
Tango	43:31	251
Waltz	37:37	217
Paso Doble	34:30	199
Quickstep	43:31	251
Rumba	39:21	227
Viennese Waltz	43:31	251

Table 2.4: Dataset statistics of our final version of *YT8M Ballroom*. This dataset was build as a subset of the collection described by the class distribution in Figure 2.3b

Chapter 3

State of the art and existing tools

We review the state of the art of dance recognition from video and the skeleton-based human action recognition with a glimpse into general human action recognition. Furthermore, we review the human pose estimation methods and we choose a particular framework for our purpose.

3.1 Review of dance recognition using visual information

Attempts at recognizing dance from videos were previously made by [13] and recently by [17]. Both of the authors were preoccupied with classifying the *Let's Dance* dataset. Unfortunately, their results are not comparable as they are each dealing with a different version of the dataset.

As for the classification method, [13] proposes using "Three-Stream Temporal CNN". This is a 3D CNN applied to three different representations of the video - the 'streams'. Results of processing individual streams are then combined together. The first of the streams uses the representation of the video as a sequence of RGB frames. The second stream works on the representation corresponding to a sequence of pictures of extracted skeletons. Note that the representation is again in the form of sequence of images - ignoring the possibility of representing the skeletons as graphs. Similarly, the third stream uses a representation of a sequence of images - this time visualizing the optical flow of the original video.

To handle the visual information, the authors of [17] use again the Three-Stream Temporal CNN. In their approach however, they replace the originally used CNN (AlexNet [32]) with a different one (Inception v3 [33]). The main contribution of this work lies in employing the audio-classification as well and thus utilizing the multimodal potential of the dance classification problem.

To the best of our knowledge, the Three-Stream Temporal CNN is thus the only relevant DNN approach tried in classification of ballroom dances. In order not to have a too narrow-sighted view of the problem, we proceed by exploring a more general field of 'Human Action Recognition'[1]. Moreover,

[1]Although our case might be considered to be more of an activity, rather than an action, most of the authors do not make the distinction between 'Human action recognition' and

our main focus lies on utilizing the skeletal representation of the dance and we consider its usage in form of images (as done by [13] and [17]) rather oversimplified.

3.2 Review of Human Action Recognition

Recognition of action from video is difficult due to the temporal natural of the problem. The temporal character effectively provides a third dimension on top of the two spatial dimension of the video frames. This increased dimensionality is the main root of the complexity of this task.

In our review, we mention two approaches to action recognition from videos. First one is to straightforwardly work with the RGB videos. This approach benefits from the possibility of seeing the context, e.g. seeing a man holding a baseball bat can hint towards actions linked with baseball bat and so on. However, the approach suffers from the above-mentioned large dimensionality.

The second approach relies on first using a human pose estimation framework on the videos to extract the location of human keypoints in each frame, these extracted keypoints of a human body are referred to as 'skeletons'. Since the frames are then represented by the skeletons only, the dimensionality gets reduced significantly. On the other hand, by only using the skeletons we inevitably lose some of the context.

3.2.1 Quo Vadis Action recognition?

The 2017 state of the art in action recognition is partially reviewed in [18]. The authors review 3 popular, distinct approaches and they introduce novel approach of their own. The point of the paper is to tackle the question whether pretraining a video classifier on a general large dataset helps with classification (after some fine-tuning) of some different, smaller dataset. The authors analyze experiments with training the four different architectures on the *Kinetics* dataset and their subsequent testing on two smaller datasets 'HMDB-51' [55] and 'UCF-101' [56]. The authors reach a conclusion that it is useful to utilize this transfer learning approach in action recognition from videos.

In the following, we mention the architectures reviewed in [18]. However we are quite brief, since none of the approaches utilizes the skeleton-based classification as we do.

ConvNet + LSTM:

The idea is to utilize 2d convolutional networks, which are so powerful in the image classification domain. However, if just the image classification was used on each frame separately and the final decision was pooled over all

'Human activity recognition' so we will stick to the more frequently used term, i.e. 'Human action recognition'.

frames, the temporal structure would be completely neglected. To this end, a recurrent layer (such as LSTM) is added to the model [18].

3D ConvNets

Convolutional networks with 3d convolutional filters applied to the stacked video frames, where the stacking along the time axis is considered as the third dimension. It is a straightforward approach, however it suffers from the large number of parameters due to the extra dimension of the filter. This makes the networks hard to train. Due to this problem they in fact fell behind the state of the art in 2017 [18].

Two-Stream Networks

These types of networks utilize convolutional layers with both RGB images and externally computed optical flows on the input. Extension of this approach uses 3d convolutional layers on top of the 3d snippets of the video (third dimension being time) [18].

Two-Stream Inflated 3D ConvNets [18]

The authors of [18] proposed an approach relying on training separately two 3d ConvNets and averaging their outputs at test time. First network was trained on the RGB inputs and the second one on the optical flow inputs. The novelty of the approach lies in utilizing successful 2d CNN architectures and using their pretrained knowledge for training of the 3D CNNs. The work [18] contains more details. Similar idea is utilized in the approach to dance classification introduced in [13].The main difference being in [13] using one more stream corresponding to the images of the skeletons.

Certainly, many novel approaches to Human Action Recognition have been introduced since 2017. To keep in line of our main topic, we further elaborate only on approaches relying on skeletal representation of the persons from videos.

3.2.2 Skeleton Based Action Recognition

We review Skeleton Based Action Recognition a bit more carefully, as we have decided to follow this approach in our work. Since we want to distinguish between particular types of dance, we believe the context might be quite similar for different classes - except perhaps for different dresses used in latin and standard dances. Losing this context might not be as painful as in other tasks.

There are several benchmarks concerning the skeleton-based action recognition [23]. Popular [19][20] are the benchmarks concerning datasets of 3d skeletons which were captured using depth sensors, such as NTU RGB+D [21] and NTU RGB+D 120 [22]. For practical purposes however, the use of 2d skeletons obtained from simple RGB videos is more promising as there is much more data available in this domain - such as the datasets introduced in Chapter 2. Therefore, we decided to concentrate on the *Kinetics-Skeleton* dataset as the most relevant benchmark for our work.

Skeleton Based Action Recognition on Kinetics-Skeleton dataset

Since there are several entries from the year 2020 tackling this benchmark [24],

We can see that tackling the *Kinetics-Skeleton* dataset is an active area of research, as there are several entries from 2020 in the benchmark [24]. Moreover, we hope we can expect even better performance on this benchmark in the coming future as the performance of the models does not seem to reach saturation[2].

As of December 2020, the models performing at the state of the art are mostly relying on Graph Convolutional Networks (GCN) which were - in the context of skeleton based action recognition - first introduced by [11] in 2018. In the rest of this section, we summarize the papers which greatly influenced the current state-of-the-art approaches to tackling the *Kinetics-Skeleton* benchmark.

Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition [11]

The work [11] introduces this - at the time novel - approach to tackle the simultaneous feature extraction throughout both the spatial and the temporal dimensions of the skeleton sequences. In the case of 2d spatial skeletons, they represent the skeleton sequence corresponding to some action as a 3d graph $G = (V, E)$, where the third dimension corresponds to time. They define the set of vertices V in (3.1):

$$V = \{v_{ti} | t = 1, ..., T, i = 1, ..., N\} \tag{3.1}$$

Where T is the number of consecutive frames covering the action, N is the number of keypoints of the skeleton in each frame and v_{it} are the coordinates of i^{th} keypoint at frame t. The set of edges E is formally defined as consisting of two kinds of edges, i.e.: $E = E_S \cup E_F$, where:

$$E_S = \{(v_{ti}, v_{tj}) | (i, j) \in H, t = 1, ..., T\} \tag{3.2}$$

The subset E_S stands for edges connecting the vertices corresponding to keypoints i and j, which form a pair of directly connected keypoints in the model of human body H and these edges are accounted for in each frame.

$$E_F = \{(v_{ti}, v_{(t+1)i}) | t = 1, ..., (T - 1), i = 1, ..., N\} \tag{3.3}$$

Where the subset E_F represents edges connecting the same keypoints accross neighboring frames.

[2]Indeed, an archive preprint [22] contributed significantly to the progress of state of the art on April 28, 2021.

Inspired by CNNs in the image recognition domain, the authors of [11] propose to use a more general version of the convolution operator which could work with spatial-temporal graphs. By this they arrive to the definition of convolutions on graphs. First, they define the spatial graph convolution (3.4) on a skeleton from a single frame, i.e. without the temporal edges:

$$f_{OUT}(v_{ti}) = \sum_{v_{tj} \in B(v_{ti})} \frac{1}{Z_{ti}(v_{tj})} f_{IN}(v_{tj}) \cdot \vec{w}(l_{ti}(v_{tj})) \tag{3.4}$$

The output of the convolution $f_{OUT}(v_{ti})$ for each vertex v_{ti} is computed using $B(v_{ti})$ - the set of the vertices in frame t which share an edge with v_{ti}. The term $f_{IN}(\cdot)$ is an input of the convolution defined over the input vertices, $Z_{ti} = |\{v_{tk}|l_{ti}(v_{tk}) = l_{ti}(v_{tj})\}|$ is the normalizing term, where $|\cdot|$ stands for cardinality of set. The mapping $l_{ti}(v_{tk})$ maps a vertex v_{tk} from the neighborhood of v_{ti} to a certain label, these labels are defined by a particular partition of the neighboring sets, see [11] for details. The term $\vec{w}(l_{ti}(v_{tj}))$ is a weight factor, which can be defined in several ways.

By extending the neighborhood B to the temporal domain as in (3.5), the generalization from the spatial convolution becomes straightforward.

$$B(v_{ti}) = \{v_{qj}|d(v_{tj}, v_{ti}) \leq K, |q-t| \leq \left\lfloor \frac{\Gamma}{2} \right\rfloor\} \tag{3.5}$$

Here, $d(v_{tj}, v_{ti})$ returns the length of the shortest path between the vertices v_{tj} and v_{ti}. K denotes the maximal spatial extend of the neighborhood and Γ is a parameter to control the temporal extend of the neighborhood.

Having redefined the problem using the graph convolutions, the authors then build a GCN, quite analogically to building a CNN. At the time, their results on the Kinetics-Skeleton dataset outperformed significantly the state of the art.

Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition [27]

Two work [27] introduces two improvements to the ST-GCN approach.

Firstly, [27] tackles the optimization of the structure of the spatial graph - claiming that having only connections between neighboring keypoints is not sufficient to capture some relevant features. For example, a correlated movement of both hands could hardly be captured by the ST-GCN model, as both hands are far apart in the graph.

Since different connections between joints are relevant for different actions, the authors argue that an adaptive approach, which can suggest relevant graph construction for different cases is necessary. To this end, an adaptive layer is introduced. This layer learns the optimal connection of the graph simultaneously during the training of the classifier.

Secondly, [27] defines a new representation of the skeleton as a directed graph with edges corresponding to the 'bones' of the skeletons. The authors argue that highlighting the orientation of the bones can be of benefit and they propose to use a second network operating on this alternative representation. The resulting two stream network thus consists of one network operating on the skeletons formed by the keypoints and the second network operating on the skeletons represented by the 'bones'. The outputs of each of the softmax layers are summed at the end of two networks and classification is performed using the added score.

Temporal Extension Module for Skeleton-Based Action Recognition [28]

The authors of [28] introduce a further improvement to the ST-GCN based approach - this time in the form of an independent module which can be placed wherever the spatial and the temporal convolutions are computed separately. In such a case, their proposed module is placed between the spatial and the temporal convolution layers and it introduces further edges along the temporal dimension. These extra edges enable for accounting for correlation between different body part movements - unlike the usual approach where the temporal edges are only connecting the same keypoint throughout time. Note the difference to [27], where the adaptive approach is taken with respect to the structure of the spatial graph, whereas the newly introduced Temporal Extension Module optimizes the temporal structure of the graph.

In case of embedding the Temporal Extension Module [28] into the Two-Stream Adaptive GCN [27], the method ranks first on the Kinetics-Skeleton dataset, according to [24] as of March 2021. Unfortunately, we could not find the code for this module online - this might be due to the authors affiliation with a private company.

Disentangling and Unifying Graph Convolutions for Skeleton-Based Action Recognition [29]

The paper [29] introduces a novel network architecture called MS-G3D. This model builds upon previous GCN-based classifiers, such as [11] and [27]. Currently - as of March 2021- MS-G3D ranks second on the Kinetics-Skeleton benchmark [24] and they are the top project which published their code, being very close in accuracy to the top performing model (38.0% vs. 38.6%). We decided to build upon their work and use MS-G3D for classification of The 10 Dances.

MS-G3D introduces two main improvements to the previous ST-GCN based approaches:

1. MS-G3D aims at capturing the cross-spacetime correlations, similarly to what the Temporal Extension Module [28] does. Example where this is useful is the action 'standing'. When standing, the body first leans forward and the legs stretch afterward [29] - this is not just a spatial or just a temporal correlation. To this end the 'G3D' module is introduced.

2. The issue of biased weightening of different walks on the graph is tackled. When we compute the neighborhood of a keypoint of size k, by taking

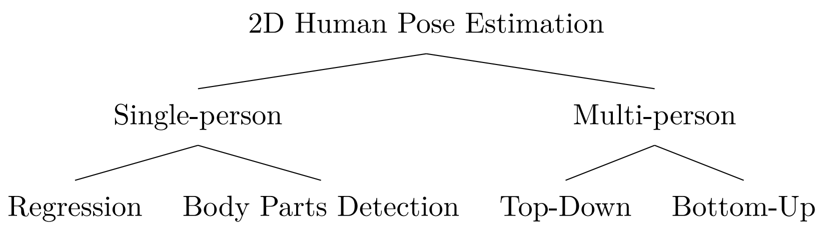


Figure 3.1: Taxonomy of 2D Human Pose Estimation [30]

k-th power of the adjacency matrix A - there will be bias towards close keypoints, because there will be short cycles which enable more walks between keypoints which are close. The paper [29] introduces a method of 'disentangling neighborhoods' which successfully tackles the biased weightening problem. The improvement makes the model 'multi-scale' - hence the name 'MS-G3D'.

3.3 Human Pose Estimation

Human Pose Estimation is an active area of research, with applications in human-computer interactions, motion analysis, virtual reality and robotics [53]. The simplest approaches rely on sensors placed onto the person of interest. However, for practical applications it is much more interesting to estimate the pose from a regular RGB video.

3D vs. 2D. While having an estimated 3d pose would be extremely useful for any application - including ours - there is a significant obstacle in achieving that. Without apriori knowledge of the scene, regular RGB single-view video is not sufficient for providing us with all the necessary information. Either a specialized camera with depth sensor (RGB-D camera) would have to be used in obtaining of the video or the scene would have to be filmed from at least two distinct viewpoints by at least 2 regular RGB cameras. Since none of the two assumptions is met in our datasets, we need to stick with 2D Human Pose Estimation in our work.

There is a large amount of different neural networks which can perform the pose estimation. For orientation in this 'model zoo', we divide the models into different groups - adopting the classification introduced in [30], see Figure 3.1.

3.3.1 Single-person pose estimation

In single person pose estimation from an image, we assume only one person is present at maximum - otherwise the image needs to be cropped until only one person is present in the snippet. There are two major approaches towards the estimation of the keypoint locations of a single person, they are called Regression and Body Part Estimation [30].

Regression

In regression, the goal is to learn all desired keypoints from images of humans at once. Pioneering in this field is the framework 'DeepPose' [31] with AlexNet [32] as a backbone. DeepPose in fact turned the attention of the whole field towards deep learning in 2014 [30].

Later improvements in regression are mainly due to better backbone architectures and the so-called multi-task learning. The idea of multi-task learning is to share feature representation between two different tasks - one being pose estimation and the other being for example action recognition or sliding-window body parts detection. Sharing of the representation leads to better generalization on the original task of pose estimation [30].

Body Parts Estimation

A different method relies on first detecting the body parts and later associating them together to form the skeleton.

The body parts are estimated one by one, where for each body part the method tries to estimate its heatmap - i.e. 2d probability distribution of the true location of the joint (the body part). To this end, the ground-truth heatmaps are mostly constructed as a 2D Gaussian centered around the ground-truth location of the joint [30].

Improved models take into account the spatial model of the body and the joint probabilities of body parts occurrence. Even later models utilize the multi-stage approach by stacking the models on top of each other, so that later models can take into account the belief maps from previous stages and improve them [30]. Recent models utilize the so-called 'stacked hourglass' architectures relying on the consecutive pooling and upsampling layers [30]. Current trends also tend to:

- Using Generative Adversarial Networks in generation of plausible whole body poses [30]
- Utilizing the temporal structure of the video and benefitting from the information stored in neighboring frames [30]. This only applies to pose estimation from video and cannot be applied in case of static images.

.

3.3.2 Multi-person pose estimation

Methods of multi-person pose estimation are of greater relevance to us as we need to estimate the pose of both dancers. This task is naturally more challenging, since it is usually not known how many persons there are in the image. Moreover, even when the number of people is known, there still remains the extra task of figuring which keypoint belongs to which person.

Top-Down approach

The basic idea of the Top-Down approach is to divide the pose estimation in two steps. Firstly, individual persons are detected and enclosed by a bounding box using a person detector. Secondly, the bounding box is fed into one of the single-person pose estimators described in 3.3.1.

Using a straightforward out-of-box person detector suffers greatly from occlusions which are no exceptions in multi-persons scenes [30]. To this end, current works concentrate on improving the detection step by various approaches - the most straightforward being the training on special datasets of occluded poses.

Drawback of the Top-Down approach lies in its speed, since the time to process an image is proportional to the number of people in that image. On the other hand, Top-Down approach is nowadays more wide-spread as it tends to be more accurate in most scenarios [35].

Bottom-Up approach

Traditional Bottom-Up approaches rely on first detecting all of the keypoints and then associating them together to form human poses. This approach was first successfully taken by DeepCut [34] [30] where the authors chose to associate the parts using integer linear programming.

Later frameworks rely on deep learning - not only in the detection but also even in the association stage - to some extent. Notably, the authors of OpenPose [36] introduce the concept of 'Part Affinity Fields' - estimated vector fields corresponding to likely position and orientation of the limbs which significantly improved the performance in the association stage.

Novel Bottom-Up approaches build upon [38], that introduced the single-stage approach in which both the body part detection and the association is learnt end-to-end by one network [30].

3.3.3 OpenPose

Exploring the competition. In choosing the pose estimation framework for our project, we tried hands-on three different frameworks: AlphaPose [40], HRNet [41] and OpenPose [36].

Due to the models being rather heavy, a good GPU is almost essential in order to run them. It makes them less accessible to install and run. Luckily, there is a GitHub project [39] gathering Google Colaboratory notebooks with installations of many deep learning frameworks including AlphaPose, HRNet and OpenPose. We provide a demo of how the three models perform on an example video of two dancers together with an informal review of some of the other frameworks in form of a Google Colaboratory notebook [50].

Reasons for OpenPose. In the end, we decided for using OpenPose despite the fact that it nowadays falls behind the state of the art in terms of accuracy, see benchmark [25]. One of the main reasons is convenience - the authors provide a well documented GitHub repository and the framework has a Python API. The second reason is that - since OpenPose up until recently had the status of the de facto standard in the field - many skeleton-based action recognition frameworks are designed to be used with skeletons corresponding to the OpenPose format. This is best witnessed by the fact that the released version of *Kinetics-Skeleton* was prepared using OpenPose.

Practical details

The official OpenPose implementation [37] is able to perform at real-time or close to real-time, but only when an appropriate GPU is available. For some simple experiments or even for processing of couple of dozens of videos, the Google Colaboratory service and its free GPUs can be used. To this end, we prepared a notebook which can be run to process a small[3] dataset and we make it available on our GitHub [50].

To process larger datasets - such as those introduced in Section 2.2 - we utilize the OpenPose Python API[4] and prepared scripts which we ran in Singularity containers using SLURM framework on the RCI cluster of CTU, see Acknowledgment.

Skeleton formats. OpenPose provides an option to generate different formats of skeletons, depending on the differently trained models. The models and corresponding formats are:

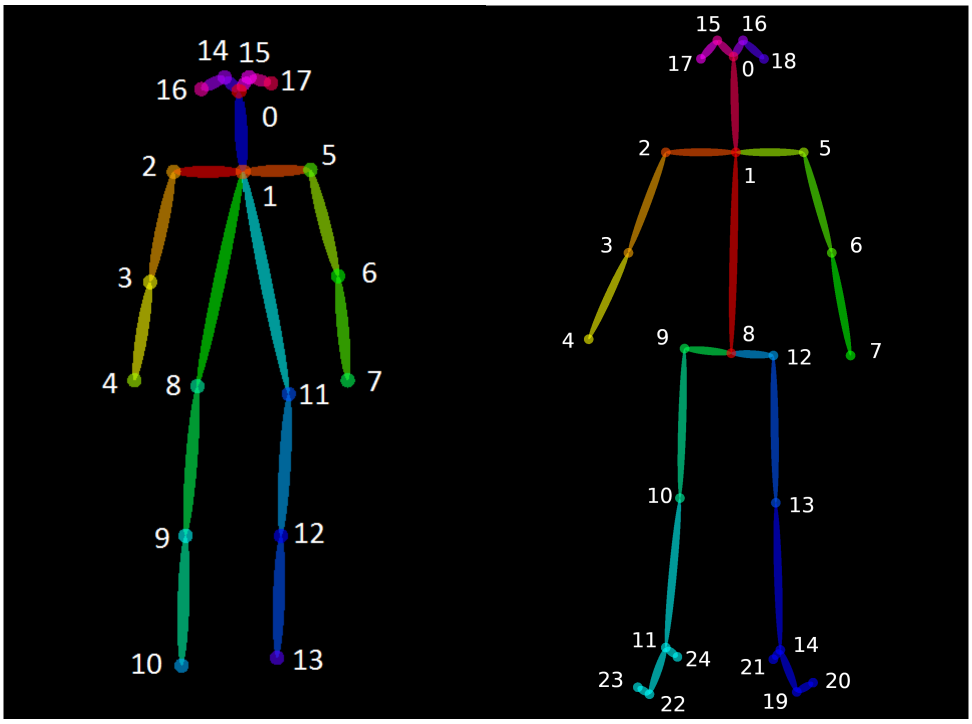
- 'COCO' - returns standard 18-keypoint skeleton format
- 'BODY_25' - most precise, fastest model returning 25 keypoints (similar to COCO + feet keypoints)
- 'MPI' and 'MPI_4_layers'- return 15 keypoints
- Other plugins can provide detailed hands and face keypoints, we do not use those

Each of the keypoints - regardless of the model - is represented as a 3-tuple of numbers. Two entries correspond to the 2d image coordinates of the detected keypoint and the third entry encodes the confidence of the detection (as a probability).

For our case the 'COCO' and 'BODY_25' models are of the greatest relevance, see Figure 3.2. The 'COCO' 18-keypoints format is the most wide-spread in 2D skeleton-based action recognition (it is the format of *Kinetics-Skeleton*) but the 'BODY_25' model performs better. We decided to use the 'BODY_25' model but later to manually drop the redundant keypoints and map the rest to the 18-keypoints format - this way we use the more precise model while preserving the compatibility with the more wide-spread skeleton format.

[3]There is a limit to the GPU usage and the runtime gets disconnected after some time.

[4]This has a significant advantage, since it enables starting OpenPose only once and then process the videos. Using OpenPose from command line does not make this possible as it reruns OpenPose for each video, taking a lot of time.



(a) : 18-keypoint skeleton format returned by the 'COCO' OpenPose model.

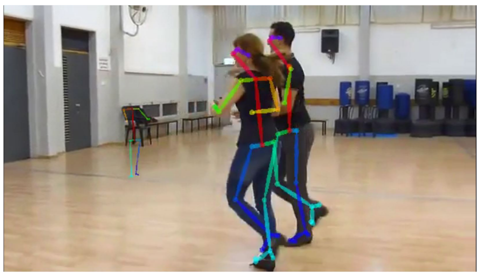
(b) : 25-keypoint skeleton format returned by the 'BODY_25' OpenPose model.

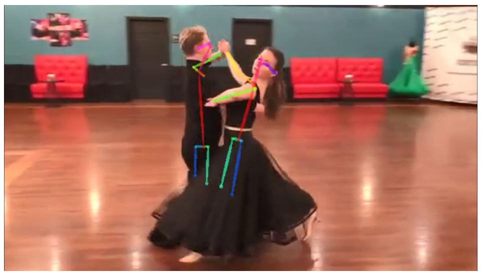
Figure 3.2: Comparison of the 18 and 25 keypoint formats. Original source of the images is not available anymore, credit likely due to [37].

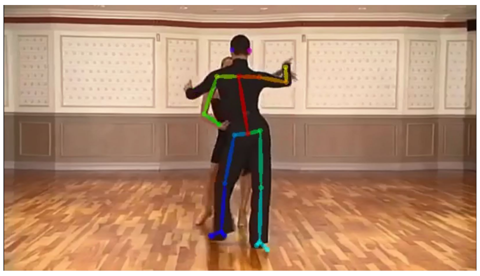
Tracking. The original version of OpenPose does not perform any tracking, i.e. for each frame it returns the skeletons of the observed people independently from the neighboring frames. However, tracking of the skeletons is essential for our purpose, as in the recognition we want to utilize the temporal structure of the motion. If two persons are detected in two consecutive frames, we want to match the skeletons from the latter frame to the corresponding skeletons in the previous frame.

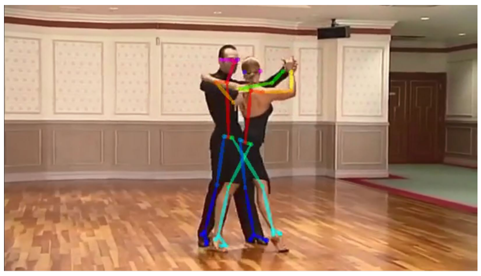
There is a later improvement to OpenPose, called 'STAF' - Spatial-Temporal Affinity Fields [42]. This work claims to solve the tracking for OpenPose while utilizing the context from neighboring frames to even improve the accuracy of the pose estimation. As a drawback, the authors admit that the STAF algorithm can fail under scene changes. Despite the claims of the OpenPose authors, this framework does not seem to have an official support from OpenPose and to the best of our knowledge it was not integrated into OpenPose. Furthermore, it does not support the 'BODY_25' model, according to the GitHub repository of STAF [43]. As such, we decided to implement our own tracking procedure instead.

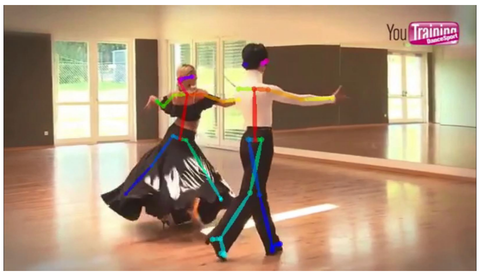
Failure examples. For illustration, we provide few examples where OpenPose partially failed on some of our data, see Figure 3.3. For more examples (incl.


(a) : Rare case of a 'ghost'. Chair with a jacket mistaken for a person.


(b) : Sizable skirts occluding legs can cause issues.


(c) : One of the dancers is almost perfectly occluded, yet her arm is assigned to the detected dancer


(d) : Body parts detection of the bottom-up approach worked well. Association stage failed by switching legs.


(e) : Body parts detection failed and proposed parts of the skirt as candidates for the knees and feet.

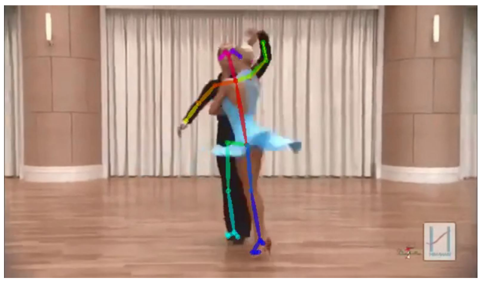

(f) : 'Frankenstein-like' error. Body parts of two different dancers mixed into a single pose.

Figure 3.3: Example situations of OpenPose errors. Generally, there are more errors with lower resolution recordings. Occlusions are challenging.

successful ones), see our GitHub [50] where we provide 10 videos with overlaid skeletons, each corresponding to a different dance style.

Chapter 4
Proposed method

We propose a GCN-based method of classifying dance videos represented by skeleton sequences. The skeleton sequences are built using a tracking procedure of our own, while utilizing OpenPose in the pose estimation stage. Furthermore, we propose several methods to normalize the skeleton sequences.

4.1 Overview of the pipeline

In Figure 4.1, we present the overview of the proposed method. The pose estimation stage is completely performed by OpenPose, using its pretrained model called BODY_25.

In the following step, we attempt to create 2 parallel skeleton sequences such that each sequence corresponds to a single dancer and we - optionally - normalize these sequences. We call this procedure 'Skeleton preprocessing', it was designed and implemented by us and it is described in Section 4.2.

Lastly, in Section 4.3 we describe how we utilize a *Kinetics-Skeleton* pretrained MS-G3D network in classification of The 10 Dances.

4.2 Skeleton preprocessing

After processing the videos with the OpenPose framework, we have a representation in form of a different set of skeletons per each frame. As described in the previous section, there is no particular ordering of this set which would induce any relation between the skeletons in consecutive frames.

For our purposes this representation is unsatisfactory, we try to improve it by designing a tool of our own. This tool firstly tries to solve the inter-frame skeleton association by performing a simple tracking. Secondly, the tool selects two skeletons out of the potentially many detected ones and sets them as the assumed dance couple. Optionally, the 'dance couple' can be mapped into a separate reference frame for normalization. To evaluate the potential contribution of the normalization step, we perform an ablation study in Section 5.1.1.

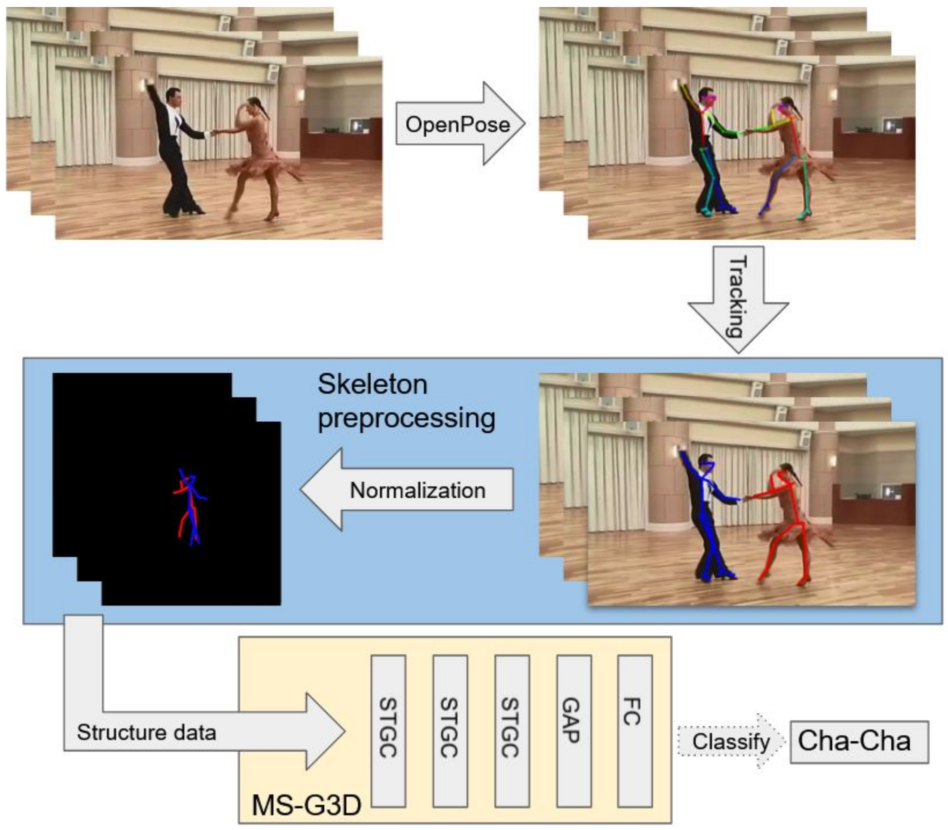


Figure 4.1: Overview of the proposed method. Firstly, videos are processed using OpenPose. Secondly, skeleton preprocessing consisting of tracking and optional normalization is performed. Lastly, MS-G3D classifier is trained on top of the skeleton sequences. Note that the skeleton drawings are solely for visualization purposes, the classifier works on graphs not on images.

4.2.1 Setting parameters of the skeleton sequences

Firstly, we set few parameters that we use to filter out some poorly detected skeletons. The values stated below, are those we use in our work. These values were not a subject of any optimization - so improvements of these parameters are likely.

- MIN_KEYPOINTS = 16 ... minimal number of keypoints detected by OpenPose (out of 25), so that the skeleton is not discarded due to incompleteness or a potential noise
- MIN_FRAMES_IN_TRACKLET = 10 ... tracking parameter - number of consecutive frames where the skeleton needs to be detected to consider the tracklet relevant
- TRACK_MAX_GAP = 5 ... tracking parameter - maximal number of frames to wait for redetection

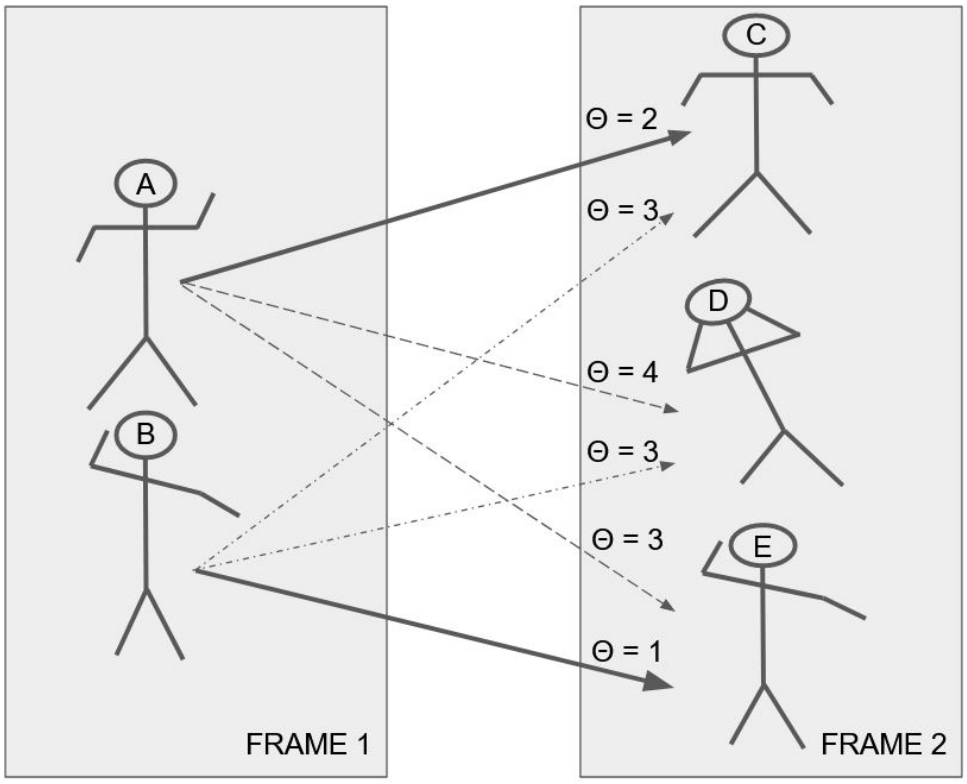


Figure 4.2: Illustrative matching of skeletons in two consecutive frames. Arrows with corresponding Θ value represent some made-up value of pseudo-distance of the two skeletons. Matching is performed in greedy manner, i.e. first the closest two skeletons are matched (here B - E) and they are removed from the matching task. Then other matches are searched for (here found A - C). In this example we have one more detected dancer in Frame 2 (Skeleton D) - this skeleton does not get matched and it is assumed to be a new person who entered the scene. In the next frame, matches to this new person will be searched for as well.

4.2.2 Tracking

The tracking procedure is based on matching of the skeletons in consecutive frames based on their 'pseudo-distance'[1], defined as:

$$\Theta(S_1, S_2) = \frac{1}{|K_1 \cap K_2|} \sum_{k \in (K_1 \cap K_2)} \|S_1(k) - S_2(k)\| \tag{4.1}$$

In (4.1) S_1 and S_2 are the 2d image coordinates of the detected keypoints for the two compared skeletons. K_1 and K_2 are the sets of the keypoints detected in the respective skeletons with non-zero confidence. The norm in the summand is Euclidean.

The matching in two consecutive frames is performed in a greedy manner, see Figure 4.2 for illustration.

[1]We use this vague term on purpose, since the function does not meet the axioms for a distance measure, such as the triangle inequality or the identity of indiscernibles.

Defining tracklets. Using the described greedy procedure, we match the consecutively detected skeletons into what we call a ’tracklet’. After processing the whole clip we are left with some tracklets, which can sometimes consist out of a few frames only - for example due to the the dancer being alternatly occluded and redetected. Using the parameter ’MIN_FRAMES_IN_TRACKLET’, the shortest of them are filtered out and dropped.

Connecting tracklets. Due to the imperfection of the pose estimation stage - mostly caused by occlusion - the major challenge of the tracking stage is to match together the tracklets corresponding to the same dancer who was not detected for few consecutive frames, but was later redetected. To this end, we introduced the ’TRACK_MAX_GAP’ parameter, which sets maximal number of frames during which an occluded skeleton can wait for redetection.

For each tracklet we build a set of tracklets within the maximally allowed gap, to which the tracklet could be connected. The final result heavily depends on the order in which the tracklets are connected, therefore we first sort them by priority. We define the priority as the duration of the tracklets in frames (we assume that the more important people in the video - hopefully the dancers - will be center of the focus of the camera).

In the order of the priority we repeatedly connect the tracklets[2] until there are no more tracklets which could be connected. We call each such chain of tracklets ’track’.

Lastly, we provide an optional step where merging of the tracks with a different value of the maximal gap can be performed - this time in the order of the priority of the tracks. The priority of tracks is defined by the number of tracklets inside the track - to account for tracklets which were repeatedly occluded during the dance but were repeatedly redetected. In our experiments however, we did not use this step.

Defining dance couple. For now, our solution is picking the representative dancers solely based on the two currently dominant tracks. When one of the tracks ends, it gets replaced by first available track. This simple approach should be enough for *Dance Tutorials Dataset*, where for the most of the time just the two dancers are in the scene and other people occur only occasionally and for a short period of time - these occurrences should not create a significant track.

The above-mentioned approach might cause problems with more populated scenes. Our hope is that even when our followed track jumps from one person to another, they should still perform the same dance (if the scene is crowded by several dance couples) and it should not matter too much to the dance recognition task. Problematic would be the case when some person from the audience gets picked as our dancer, however most of the audience is sitting and not visible well - if detected by OpenPose the majority of spectators should be filtered out based on the conditions of the minimal number of detected keypoints or the minimal tracklet length.

[2]Based on the pseudo-distance between the last skeleton in the former tracklet and first skeleton in the latter tracklet

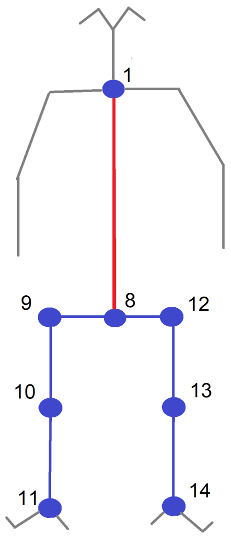


Figure 4.3: The red and blue bones belong to the set B. We call the red bone 'spine'.

4.2.3 Normalization

We propose a method of skeleton sequence normalization. The idea of this step is to get rid of unimportant information contained in the unnormalized skeleton sequences. However, we do not have a prior knowledge about which of the information contained in the sequences is important and which is not. To address this question we try out several variants described below and we compare their performance in Section 5.1.1.

Firstly, we suppress the differences between skeleton dance sequences based on diverse skeleton sizes - caused by dancers either being of different height or having different distance to camera. Secondly, we prevent the classifier from paying attention to the exact position of the skeleton within the frame by mapping each dancer into its own frame of reference.

The rescaling is challenging because we are working with 2d skeletons only. As these skeletons are only planar projections of the 'true' 3d skeletons, we lose the information about the true sizes of bones[3].

For later use, let us define a particular set of bones - we will call this set B. The set B contains a particular subset of bones of the BODY_25 skeleton. The subset is highlighted in Figure 4.3.

As a reference length for each skeleton, we decide to take the length of its 'spine' (i.e. the distance between keypoints 1 and 8 - as defined in Figures 3.2b and 4.3). If a dancer bows towards or away from the camera, due to the 3d$\rightarrow$2d projection the length of his 'spine' and thus the whole scale of the skeleton would undesirably shorten. We tackle this problem with three simplifying assumptions - the so-called *Scene Assumptions*:

[3]By bone we mean a line segment connecting two neighboring keypoints.

1. For the bones inside the set B (see Figure 4.3), it holds that the dancer is frequently in such positions where the preimage of the bones is parallel to the image plane.

2. In each frame the preimage of at least one bone from the set B is parallel to the image plane.

3. The relative distance of the dancer to the camera does not change too rapidly - i.e. the dancer is filmed from a reasonably far distance.

Defining the tracklet scale and the frame scale. Using *Scene Assumptions* we set the scale of the dancer (per tracklet) followingly:

$$s_t(T) = \max_{f \in T} \frac{l_f}{C} \tag{4.2}$$

Where the tracklet T is a sequence of frames with skeletons $\{f_1, ..., f_{|T|}\}$, l_f is the length of the 'spine' of the skeleton in frame f. The constant C is not important as it is the same for all skeletons, but it can help with proper visualization of the dancer within the bounds of the new reference frame.

By finding the maximum of the spine length we assume to have found the true length of the spine when the dancer is the closest to the camera (using *Scene Assumptions* 1 and 3).

To be able to propagate this information about the dancers scale to other frames - which might be taken at times when the distance to the camera is different - we propose to account for the changing distance to the camera. To this end, we introduce a new scale - this time not for the whole tracklet but for each frame (skeleton) of the tracklet:

$$s_f(f, T) = \max_{b(f) \in B; B \subset f} \frac{\|b(f)\|}{b_{max}(T)} \quad ; \quad b_{max}(T) = \max_{f' \in T; b(f') \in B; B \subset f'} \|b(f')\| \tag{4.3}$$

Here $b(f)$ is the bone b in frame f and $b_{max}(T)$ is the maximal length of the bone b observed within tracklet T. Where we only consider the bones from the set B, as for them we have *Scene Assumption* 1.

The term $\frac{\|b(f)\|}{b_{max}(T)}$ corresponds to the relative observed size of the bone b in frame f w.r.t. the maximal observed size of the bone b within the tracklet. The biggest relative size is met by the bone fulfilling *Scene Assumption* 3 in frame f. Therefore, we maximize this relative size to get a rough estimate of the ratio of the sizes in this frame w.r.t. the maximal sizes observed in the tracklet. This approximates the scale change in each frame induced by the varying distance of the dancer to the camera.

Normalization variants

We propose to try out three different variants of the normalization step with fourth variant as a benchmark corresponding to no normalization being performed. To this end, we first introduce the building blocks of the normalization step. Secondly, we provide the definition of the particular normalization variants at the end of this section.

Defining new reference frame. The skeletons can be obtained from videos with various resolutions. Since the unit for the skeleton keypoints remains 1px, different video resolutions would influence the length of the bones. Therefore, we define the 'resolution' of the new reference frame which we construct for each dancer - we adopt the resolution used in the preparation of *Kinetics-Skeleton* 340x256 [11].

Using OpenCV [45] we get the resolutions of the videos in our dataset and we adjust the units of the new reference frames so that it transforms the effective resolution to the required one of 340x256 pixels. This step is performed in all of the nontrivial normalization variants.

Skeleton normalization by translation. In normalization variants containing the normalization w.r.t. translation, we start the transformation of each skeleton inside the new reference frame by translating its keypoint number 1 (see Figure 3.2b) into the middle of this new frame.

In normalization variants without translational normalization, we translate the keypoint to a position corresponding to its position in the original frame.

Skeleton normalization by scale. Starting from the keypoint number 1, we construct the normalized skeleton by adding neighboring keypoints. To this end we take the bone from the original skeleton and interpret it as a vector pointing from the keypoint which was already processed to the new keypoint. In case of processing frame f of tracklet T, we multiply the bone vector by the scaling factor $\eta_1(f,T)$ or $\eta_2(T)$ - depending on the normalization variant:

$$\eta_1(f,T) = \frac{1}{s_f(f,T)\cdot s_t(T)} \quad ; \quad \eta_2(T) = \frac{1}{s_t(T)} \tag{4.4}$$

We add this new normalized bone to the corresponding keypoint in the new reference frame by which we transfer one more keypoint. We repeat the process until the whole normalized skeleton is constructed.

Normalization variants definition. For systematic evaluation of the normalization step in Section 5.1.1, we propose the following variants:

1. Full normalization - perform both the translational normalization and the frame-level scaling using the factor η_1 - see Equation (4.4). In Figure 4.4, a visualization of a fully normalized dance couple is provided.

2. No frame-level normalization - perform the translational normalization with skeleton scale normalized only on the tracklet level using the factor η_2 - see Equation (4.4)

3. No translational and frame-level normalization - perform only the tracklet level scale normalization using the factor η_2

4. No normalization at all

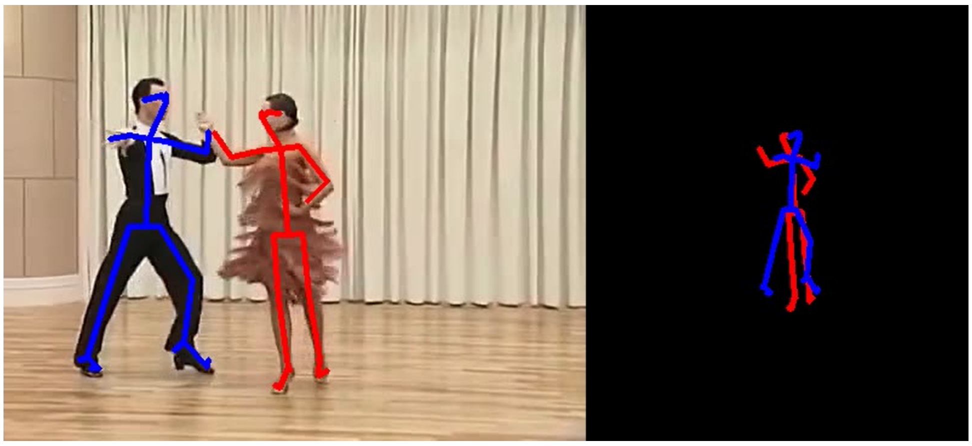


Figure 4.4: Visualization of the skeleton normalization. To the left are the dancers performing Cha-Cha with OpenPose skeletons overlaid. To the right is the new reference frame with both translationally and scale normalized skeletons. We provide a whole example video which illustrates the tracking as well, see GitHub [50].

4.3 The Classifier

We use MS-G3D - a state-of-the-art Graph Convolutional Network for skeleton-based action recognition. We take a model pretrained on *Kinetics-Skeleton* and using transfer learning techniques [47] we train it for classification of The 10 Dances.

4.3.1 MS-G3D

In Figure 4.5 the architecture of MS-G3D [29] is provided. The ideas behind the network are mentioned in Section 3.2.2.

We use a model pretrained on *Kinetics-Skeleton*, where it achieves Top-1 accuracy of 36.8% [29] using the joint-stream only. When using also the bone-stream and combining the results of both streams, the authors achieve Top-1 accuracy 38.0% [29]. We downloaded the *Kinetics-Skeleton* dataset and both of the pretrained models - one model for each stream - and we reproduced both of the reported accuracies. By default however, we only work with the joint stream - unless stated otherwise.

4.3.2 Data format for MS-G3D

Authors of [29] provide a script for transformation of the data from the *Kinetics-Skeleton* dataset into a format suitable for MS-G3D. Each *Kinetics-Skeleton* instance is represented as a JSON file, see template in Figure A.1. The provided script transforms the json files into NumPy array [46] with the shape of (N, C, T, V, M), where:

- N - The number of T-frames long clips in the whole dataset. In case of *Kinetics-Skeleton* T-frames long clip corresponds to a single instance. In case of *Dance Tutorials Dataset*, the longer instances are cut to form these clips.

- $C = 3$ - The number of channels - i.e.: 2 channels for the 2 image coordinates and 1 channel for the confidence score returned by OpenPose.
- $T = 300$ - The number of frames of the input sequences, 300 is the default value for *Kinetics-Skeleton*.
- $V = 18$ - The number of keypoints (joints) of the skeletons. *Kinetics-Skeleton* uses the 18-joint format returned by OpenPose.
- $M = 2$ - The number of persons to account for in each frame. Default for *Kinetics-Skeleton* is 2, which is useful for the case of a dance couple as well.

Evaluation of 2 skeletons per frame. The network considers each person in the frame independently of others. It pools the decision about the action being performed by adding the softmax scores of each of the skeleton sequences.

In our case we work with 2 persons per frame - the supposed dance couple. If there is only one or none skeleton detected in the frame, it is replaced by a zero array in the MS-G3D data transformation script.

Splitting longer instances into 300-frames long segments. Instances longer than 300 frames - which is the typical case of a *Dance Tutorials Dataset* instance - were split into non-overlapping 300-frames long segments. The remaining segment of the instance shorter than 300 frames was dropped. Another option would be to pad this segment for example by repeating the frames until there are 300 of them.

4.3.3 Splitting data into training and validation sets

Dance Tutorials Dataset. We split *Dance Tutorials Dataset* manually so that roughly 80% goes to the training set and around 20% to the validation set. The reason for not splitting completely at random is the varying length of the instances. We do not want to put different 300-frames segment from the same instance into both the training and the validation set. We provide the list of the instances assigned to validation set on GitHub [49].

10-Let's Dance. We perform a random 80-20 split with a fixed random seed. The list of the instances assigned to the validation set is provided on our GitHub [49].

YT8M Ballroom. We do not perform any split of the *YT8M Ballroom* dataset. Since its labels are noisy, we only use this dataset in enlarging a training set of a different dataset - keeping the validation set based only on verified labels.

Depth	Layers trained
Depth 1	FC + TCN3 + SGCN3
Depth 2	FC + TCN3 + SGCN3 + GCN3D3
Depth 3	FC + TCN3 + SGCN3 + GCN3D3 + TCN2
Depth 4	FC + TCN3 + SGCN3 + GCN3D3 + TCN2 + SGCN2
Depth 5	FC + TCN3 + SGCN3 + GCN3D3 + TCN2 + SGCN2 + GCN3D2

Table 4.1: Different sets of layers to be trained jointly. See Figure 4.5 for overview of the layers. The trailing number means the number of the STGC block - number 1 comes right after input and number 3 comes before the FC layer.

4.3.4 Transfer Learning

The MS-G3D architecture has almost 3 million trainable parameters. To be able to train this network with our dance datasets of limited size, we utilize the transfer learning approach [48][47]. We take a model pretrained on *Kinetics-Skeleton*, we replace the classification layer to match The 10 Dances and we train the classifier together with some of the original layers.

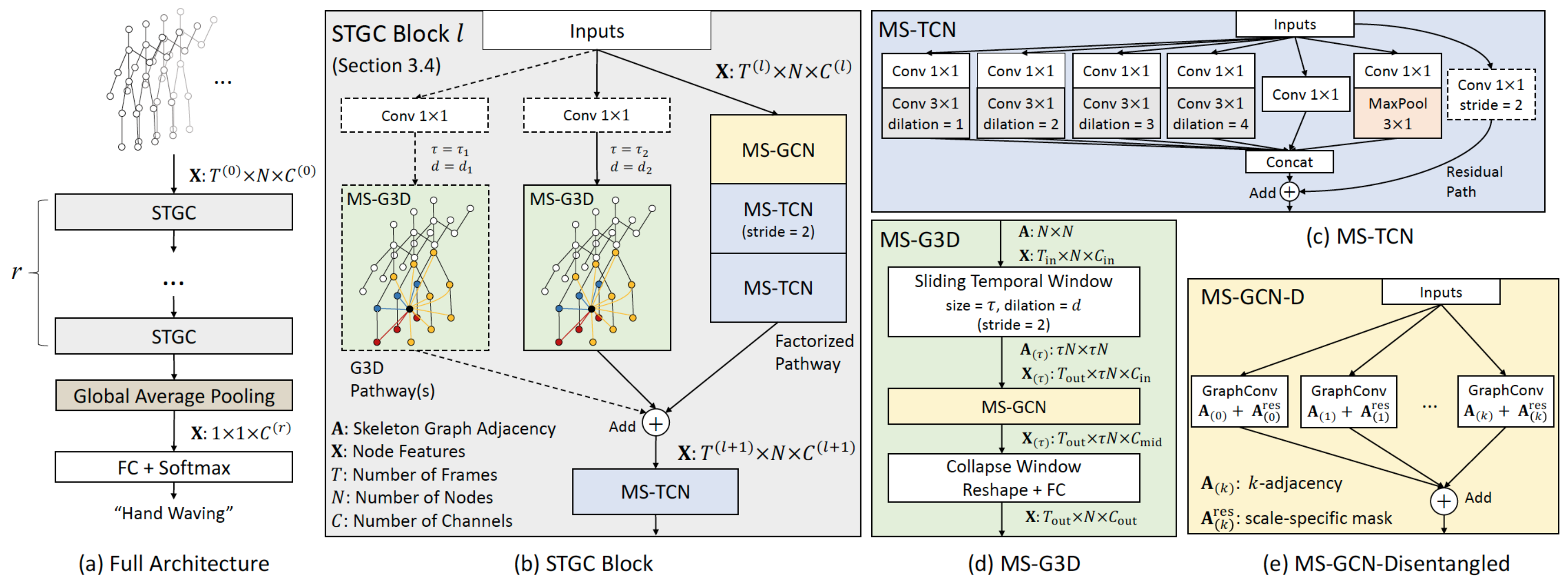
Replacing the classification layer. The final layer of the model pretrained on *Kinetics-Skeleton* is a fully connected (FC) layer with 400 output neurons corresponding to 400 classes of *Kinetics-Skeleton*. We replace this final layer by a different FC layer with only 10 output neurons corresponding to The 10 Dances. Another option would be to replace the original FC layer by a more sophisticated classifier consisting of several layers.

Since this layer is new, it does not have any pretrained parameters and it has to be trained from scratch.

The depth of retraining. We further decide which of the original layers to freeze[4] and which to train.

Keeping all layers of the original network frozen would be equivalent to using the network as a feature extractor with only the classifier trained on top of them. In Table 4.1 we propose different depths of training for systematical evaluation. However, this list is not exhaustive as there are many possible combinations of layers which could be trained jointly.

[4]Freezing means keeping the parameters of the layer fixed with the pretrained values and not changing them in the training.



$\mathbf{X}: T^{(0)} \times N \times C^{(0)}$
STGC
...
STGC
r
Global Average Pooling
$\mathbf{X}: 1 \times 1 \times C^{(r)}$
FC + Softmax
"Hand Waving"
(a) Full Architecture
STGC Block l
(Section 3.4)
Inputs
$\mathbf{X}: T^{(l)} \times N \times C^{(l)}$
Conv 1×1
Conv 1×1
$\tau = \tau_1$ $d = d_1$
$\tau = \tau_2$ $d = d_2$
MS-G3D
MS-G3D
MS-GCN
MS-TCN (stride = 2)
MS-TCN
Factorized Pathway
G3D Pathway(s)
Add
$\mathbf{X}: T^{(l+1)} \times N \times C^{(l+1)}$
MS-TCN
$\mathbf{A}$: Skeleton Graph Adjacency
$\mathbf{X}$: Node Features
T: Number of Frames
N: Number of Nodes
C: Number of Channels
(b) STGC Block
MS-TCN
Inputs
Conv 1×1
Conv 3×1 dilation = 1
Conv 1×1
Conv 3×1 dilation = 2
Conv 1×1
Conv 3×1 dilation = 3
Conv 1×1
Conv 3×1 dilation = 4
Conv 1×1
Conv 1×1
MaxPool 3×1
Conv 1×1 stride = 2
Concat
Add
Residual Path
(c) MS-TCN
MS-G3D
$\mathbf{A}: N \times N$
$\mathbf{X}: T_{\text{in}} \times N \times C_{\text{in}}$
Sliding Temporal Window
size = τ, dilation = d
(stride = 2)
$\mathbf{A}_{(\tau)}: \tau N \times \tau N$
$\mathbf{X}_{(\tau)}: T_{\text{out}} \times \tau N \times C_{\text{in}}$
MS-GCN
$\mathbf{X}_{(\tau)}: T_{\text{out}} \times \tau N \times C_{\text{mid}}$
Collapse Window
Reshape + FC
$\mathbf{X}: T_{\text{out}} \times N \times C_{\text{out}}$
(d) MS-G3D
MS-GCN-D
Inputs
GraphConv $\mathbf{A}_{(0)} + \mathbf{A}^{\text{res}}_{(0)}$
GraphConv $\mathbf{A}_{(1)} + \mathbf{A}^{\text{res}}_{(1)}$
...
GraphConv $\mathbf{A}_{(k)} + \mathbf{A}^{\text{res}}_{(k)}$
Add
$\mathbf{A}_{(k)}$: k-adjacency
$\mathbf{A}^{\text{res}}_{(k)}$: scale-specific mask
(e) MS-GCN-Disentangled

Figure 4.5: Overview of MS-G3D architecture. The particular model used has $r = 3$. TCN blocks perform the temporal convolutions, GCN blocks perform the spatial convolution, 'MS' stands for multi-scale [29]. Credit for the image is due to [29].

)

Chapter 5

Results

In this chapter, we summarize the results achieved using the method proposed in Chapter 4 on the datasets introduced in Section 2.2.

5.1 Training and evaluation on fixed size video segments

The classifier introduced in Section 4.3 is designed to work with skeleton sequences corresponding to 300 frames of video. To this end, we divide our instances into segments of 300 frames in a following way:

1. Each instance is cut into non-overlapping segments corresponding to 300 frames of the original video.
2. The last segment which is shorter than 300 frames is dropped but only under the condition that it is not the only segment of the instance
3. If the instance itself is shorter than 300 frames, it is padded by repeating the frames from the beginning of the instance

5.1.1 Models trained on Dance Tutorials Dataset

We decided to use *Dance Tutorials Dataset* for a systematic evaluation of the various depths of retraining defined in Table 4.1 as well as of the different normalization variants proposed in Section 4.2.3. The reason is that *Dance Tutorials Dataset* depicts the dance the clearest and the performance on this dataset should not be negatively influenced by other factors such as whether we keep the focus on the tracks of the dancers or on some other people - as other people are mostly not present in the videos from *Dance Tutorials Dataset*.

Ablation study of the normalization

We trained the classifier using four different variants of normalization introduced in Section 4.2.3 in order to decide which of the variants leads to the best trained model.

Depths	Base learning rates
Depth 1	{0.005;0.01;0.05;0.1}
Depth 2	{0.005;0.01;0.05;0.1}
Depth 3	{0.005;0.01;0.05;0.1}
Depth 4	{0.001;0.005;0.01;0.05}
Depth 5	{0.001;0.005;0.01;0.05}

Table 5.1: Base learning rates proposed for various training depths defined in Table 4.1

	Full norm.	**No Frame-level Norm.**	**No Transl. Norm.**	**No Norm.**
Training depth	4	1	3	1
Base learn. rate	0.005	0.1	0.01	0.01
Best epoch	42	58	9	50
Top-1 accuracy	60.8 %	60.0 %	64.8 %	**64.8 %**
Top-2 accuracy	71.2 %	69.2 %	75.2 %	**76.0 %**

Table 5.2: Accuracy of the best models depending on the normalization of the dataset. We provide the depth of retraining, the base learning rate and the epoch in which the model reached the best accuracy on the validation set.

Training hyperparameters. For most of the hyperparameters we followed those used in the training of MS-G3D on *Kinetics-Skeleton* [52]- we trained for 65 epochs, using 2 GPUs, with fixed batch size of 32 and using SGD optimizer with a weight decay of 0.0003.

As for the base learning rate, we tried different values based on the depth of retraining, see Table 5.1.

Training results. The classifier was trained in 20 different ways[1] for each of the 4 normalization variants. For complete overview of the achieved accuracies see Table A.1. In Table 5.2 we provide the best achieved accuracy for each of the normalization variants.

Discussion of the failure of the normalization procedure. The two normalization variants which include the translational normalization performed the worst. We argue that by losing the information about the dancers motion across the scene, we lose some of the features which are relevant for dance recognition.

The normalization variant which only performed the normalization with respect to the dancers size performed comparably to not using any normalization at all.

We conclude that the normalization techniques we introduced bring no improvement in the training of the model. Therefore, we decided to work only with the unnormalized skeleton sequences.

[1]Using the 5 different depths of retraining, each trained with 4 different base learning rates

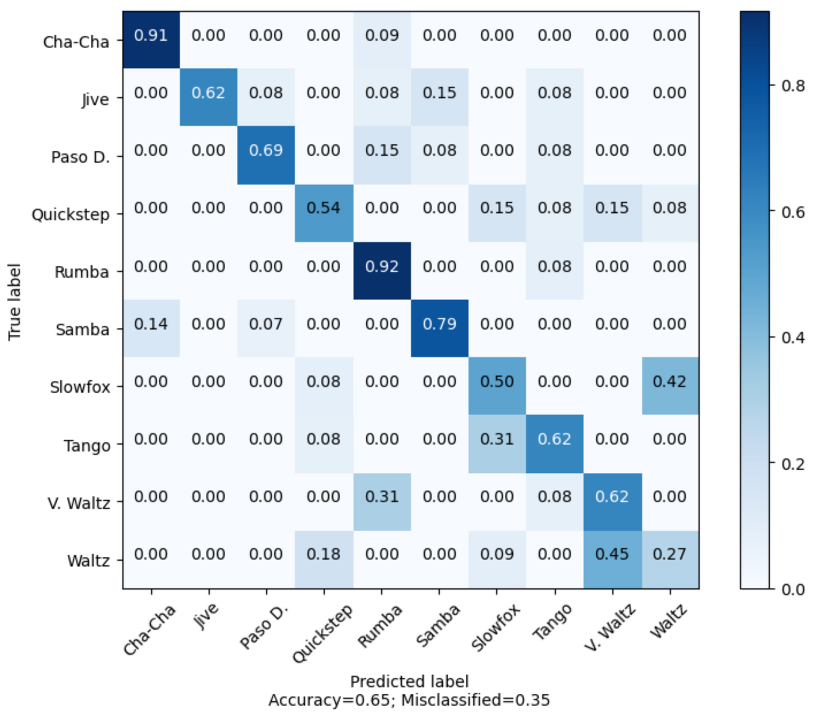


Figure 5.1: Confusion matrix evaluated using the best performing model on the 300 frames segments from the validation set of *Dance Tutorials Dataset*

The best performing model

In Table 5.2 we can see that the best model trained on *Dance Tutorials Dataset* achieved Top-1 accuracy of 64.8%. In Figure 5.1, we provide a confusion matrix summarizing the performance of the model on individual classes. Note, that the model performs better on latin dances than on standard dances[2].

We choose this model as the most representative one and we continue studying its performance in Sections 5.1.4, 5.2 and 5.3. For simplicity we further refer to this model as *The Tutorials Model*.

2-streams network

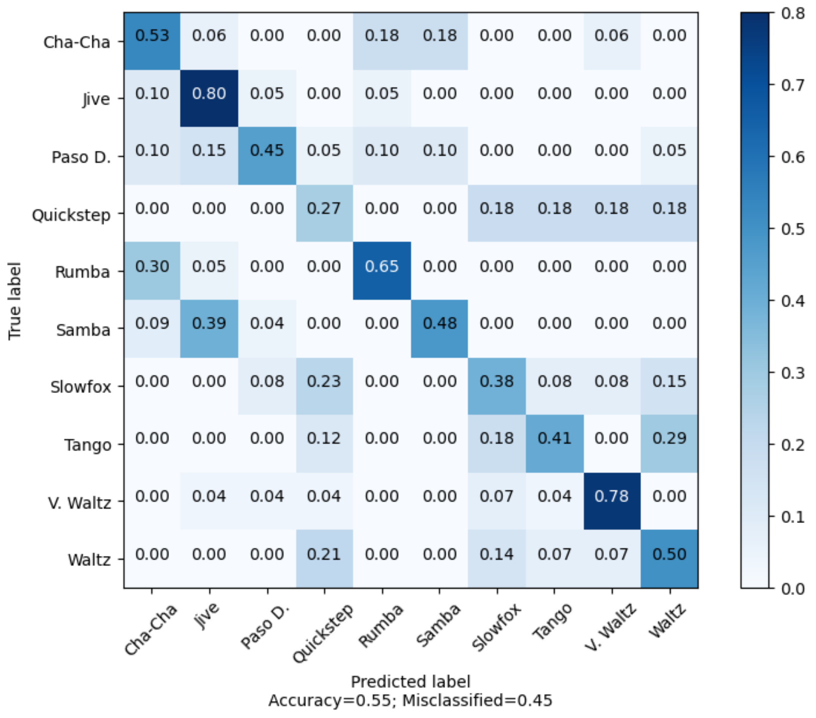
We tried out the idea of using the two stream network as in [29]. To this end, we performed 8 different trainings using the bone-stream. We provide the summary of these trainings in A.2. The best performing model relying on the bone-stream achieves Top-1 accuracy of 58.4%.

When combining the best bone-stream model with The Tutorials Model (working on the joint-stream) in the 2-stream manner we obtained Top-1 accuracy of 63.2 %, i.e. worse than the joint-stream model achieves alone. We conclude that the 2-stream network did not bring any improvement in our case, although performing more thorough hyperparameter tuning of the bone-stream network training could improve the 2-stream accuracy.

[2]For distinction between latin and standard dances see Table 1.1

Base learning rate:	**0.005**	**0.01**	**0.05**	**0.1**
Best epoch	35	39	35	40
Top-1 Acc.	52.7 %	**55.5 %**	41.2 %	28.0 %
Top-2 Acc.	75.3 %	73.6%	57.1 %	42.9 %

Table 5.3: Overview of the accuracies achieved by 4 different trainings on *10-Let's Dance*



	Cha-Cha	Jive	Paso D.	Quickstep	Rumba	Samba	Slowfox	Tango	V. Waltz	Waltz
Cha-Cha	0.53	0.06	0.00	0.00	0.18	0.18	0.00	0.00	0.06	0.00
Jive	0.10	0.80	0.05	0.00	0.05	0.00	0.00	0.00	0.00	0.00
Paso D.	0.10	0.15	0.45	0.05	0.10	0.10	0.00	0.00	0.00	0.05
Quickstep	0.00	0.00	0.00	0.27	0.00	0.00	0.18	0.18	0.18	0.18
Rumba	0.30	0.05	0.00	0.00	0.65	0.00	0.00	0.00	0.00	0.00
Samba	0.09	0.39	0.04	0.00	0.00	0.48	0.00	0.00	0.00	0.00
Slowfox	0.00	0.00	0.08	0.23	0.00	0.00	0.38	0.08	0.08	0.15
Tango	0.00	0.00	0.00	0.12	0.00	0.00	0.18	0.41	0.00	0.29
V. Waltz	0.00	0.04	0.04	0.04	0.00	0.00	0.07	0.04	0.78	0.00
Waltz	0.00	0.00	0.00	0.21	0.00	0.00	0.14	0.07	0.07	0.50

Figure 5.2: Confusion matrix evaluated using the best performing model on the 300 frames longs segments from the validation set of *10-Let's Dance*

5.1.2 Models trained on 10-Let's Dance

We trained models specifically for classification of the *10-Let's Dance* dataset.

Training hyperparameters. We fixed the depth of retraining to be 'Depth 1' as it performed the best on *Dance Tutorials Dataset*. Using the unnormalized data we trained with 4 different base learning rates keeping the rest of the hyperparameters same as in previous trainings. We summarized the results of the trainings in Table 5.3.

Training results. We can see that the best model achieved accuracy of 55.5%. The fact that the performance on *10-Let's Dance* is worse than on *Dance Tutorials Dataset* fulfills our expectation of *10-Let's Dance* being more challenging for classification.

From the confusion matrix provided in Figure 5.2, we can see that the model performed the best on the classes corresponding to Rumba and Viennese

Base learning rate:	**0.001**	**0.003**	**0.005**	**0.01**	**0.05**	**0.1**
Best epoch	34	70	59	47	63	31
Top-1 Acc.	56.8 %	64.8 %	**64.8 %**	62.4 %	52.8 %	25.6 %
Top-2 Acc.	76.8 %	75.2 %	**77.6 %**	73.6 %	69.6 %	44.8 %

Table 5.4: Overview of the accuracies achieved by 6 different trainings on *Dance Tutorials Dataset* with training set boosted by including the *Youtube Ballroom* videos.

Waltz. As Viennese Waltz was the only class of the dataset collected by us, we suspect the performance on this class might be partially due to our influence in the data collection stage. We might have been collecting the Viennese Waltz videos according to different standards than the collection of the remaining classes obeyed.

5.1.3 Utilizing YT8M Ballroom in training

We added the *YT8M Ballroom* videos together with the training set of *Dance Tutorials Dataset* to form a new training set. Using the unchanged validation set of *Dance Tutorials Dataset*, we trained a new model.

Training Hyperparameters. We performed the trainings with depth of retraining set to 'Depth 1'. Using the unnormalized data we performed 6 trainings, using SGD optimizer and weight decay 0.0003, we let the model train with learning rates 0.001 and 0.003 for 100 epochs and with learning rates 0.005, 0.01, 0.05 and 0.1 for 65 epochs.

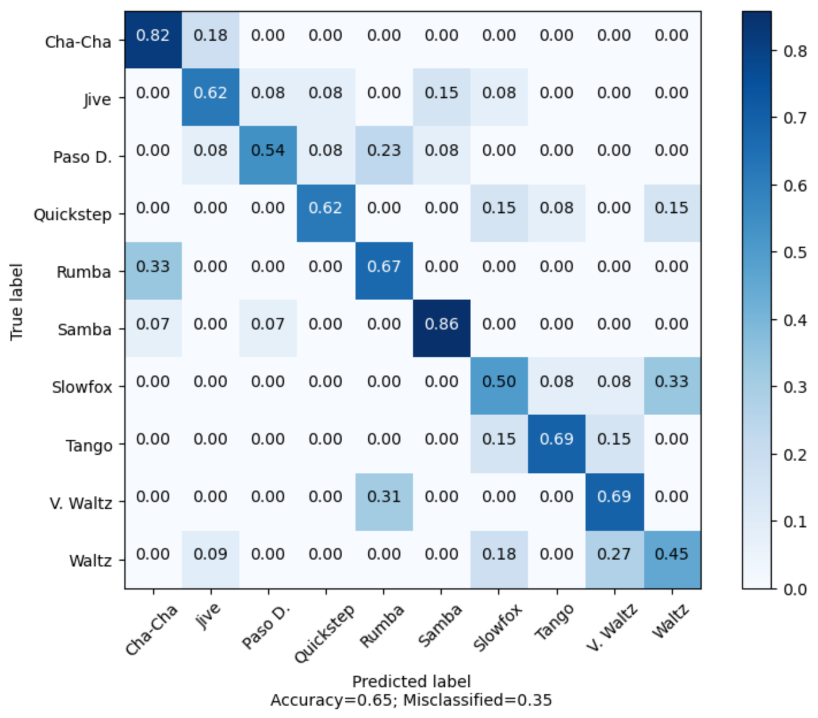
Training results. See Table 5.4 for an overview of the achieved accuracies. We can see we were able to obtain the Top-1 accuracy of 64.6%, i.e. the same as without the usage of *YT8M Ballroom*. We were even able to achieve slightly better Top-2 accuracy - this difference is however too small to be consider significant in the presence of different factors influencing the trainings - such as the choice of hyperparameters. We further refer to the best model trained using the extra *YT8M Ballroom* data as *YT8M Model*.

In Figure 5.3, see the confusion matrix obtained with *YT8M Model* on the *Dance Tutorials Dataset* validation set. In comparison with Figure 5.1, we can see a lot of similarity but we can also see that the off-diagonal elements in Figure 5.3 are distributed slightly more homogeneously. Based on this observation, we claim that the errors made by *YT8M Model* are less systematic and more random.

5.1.4 Evaluation of the models on 10-Let's Dance

We evaluated *The Tutorials Model* and *YT8M Model* on the validation set of *10-Let's Dance*, see Table 5.5. The obtained accuracies are very low[3], much

[3]Note that there is a small overlap between the datasets, which actually even improves the accuracy a little bit. *YT8M Ballroom* contains 1 video from the *10-Let's Dance* validation



True label	Cha-Cha	Jive	Paso D.	Quickstep	Rumba	Samba	Slowfox	Tango	V. Waltz	Waltz
Cha-Cha	0.82	0.18	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Jive	0.00	0.62	0.08	0.08	0.00	0.15	0.08	0.00	0.00	0.00
Paso D.	0.00	0.08	0.54	0.08	0.23	0.08	0.00	0.00	0.00	0.00
Quickstep	0.00	0.00	0.00	0.62	0.00	0.00	0.15	0.08	0.00	0.15
Rumba	0.33	0.00	0.00	0.00	0.67	0.00	0.00	0.00	0.00	0.00
Samba	0.07	0.00	0.07	0.00	0.00	0.86	0.00	0.00	0.00	0.00
Slowfox	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.08	0.08	0.33
Tango	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.69	0.15	0.00
V. Waltz	0.00	0.00	0.00	0.00	0.31	0.00	0.00	0.00	0.69	0.00
Waltz	0.00	0.09	0.00	0.00	0.00	0.00	0.18	0.00	0.27	0.45

Figure 5.3: Confusion matrix evaluated using the model trained on *Dance Tutorials Dataset* enhanced by extra training data corresponding to the *YT8M Ballroom* videos.

Model:	**YT8M Model**	**The Tutorials Model**
Top-1 Acc.	22.0 %	18.1 %
Top-2 Acc.	40.1 %	33.0 %

Table 5.5: Classification accuracy on *10-Let's Dance* validation set using the models trained on *Dance Tutorials Dataset* and *Dance Tutorials Dataset* + *YT8M Ballroom* respectively.

lower than the 55.5% Top-1 accuracy obtained when training on the *10-Let's Dance* training set (see Table 5.3). This could be explained by the different nature of the two datasets and by the low generalization ability of *YT8M Model* and *The Tutorials Model*.

5.2 Evaluation on videos of varying duration

In this section, we assess how the classification accuracy changes when we process segments longer than the 300 frame[4] ones. To this end, we use *The Tutorials Model* as the classifier and the *Dance Tutorials Dataset* validation set as the data. Note that the *Dance Tutorials Dataset* validation set consists of 36 instances in total, with each genre being represented by 3 or 4 instances.

set and *Dance Tutorials Dataset* contains 2 of the *10-Let's Dance* validation instances.

[4]In other words approx. 10 second long clips.

Model:	**The Tutorials Model**
Top-1 Acc.	72.2 %
Top-2 Acc.	86.1 %

Table 5.6: Classification accuracy on whole instances of the *Dance Tutorials Dataset* validation set, using *The Tutorials Model* in sliding window manner with window size of 300 frames and a 60 frame stride.

To process videos longer than 300 frames, we perform the evaluation in a sliding window fashion.

Sliding window evaluation. We take a 300 frames long window which we slide along each instance with a stride corresponding to 60 frames. This procedure produces overlapping segments which are fit to be processed by *The Tutorials Model*. For each such segment, the model returns a prediction in form of a probability distribution[5] over The 10 Dances. For each of the instances we add together all of the returned probability distribution and report the label with the greatest sum of probabilities over all of the segments corresponding to that instance.

Evaluation on Dance Tutorials Dataset using whole instances. By evaluating the whole instances of *Dance Tutorials Dataset* in sliding window manner, the classification accuracy notably improves - see Table 5.6. This advance is largely due to an improved classification of the latin dances, as we can see from the confusion matrix in Figure 5.4.

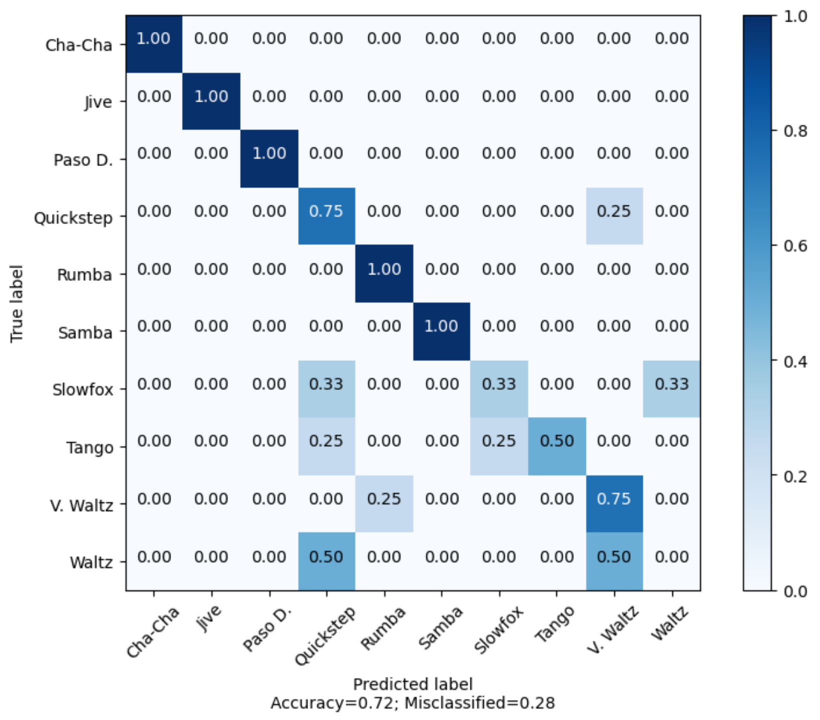
Classification accuracy and the sample duration. We evaluate how the classification accuracy of *The Tutorials Model* changes with absolute duration of classified samples. The procedure is similar to the evaluation on whole samples, only this time we report the accuracy every time a new window is evaluated - not only for the whole sample[6].

We present the dependency of accuracy on the classified segment duration in Figure 5.5. The accuracies for the longer samples are less reliable as the instances of the dataset have varying durations and for longer durations there is less data available.

In Figure 5.5, we can see that for instances with minimal duration of 30-35 seconds we were able to achieve accuracy above 80.0%, with around half of the validation set (i.e. around 18 videos) supporting the reliability of the accuracy computation.

[5]Using softmax for normalization of logits.

[6]I.e. for each instance we do not report just one label, but we report the running score every 60 frames after the initial 300 frame segment is processed. We average these accuracies at each time step over all instances with sufficient duration.



	Cha-Cha	Jive	Paso D.	Quickstep	Rumba	Samba	Slowfox	Tango	V. Waltz	Waltz
Cha-Cha	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Jive	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Paso D.	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Quickstep	0.00	0.00	0.00	0.75	0.00	0.00	0.00	0.00	0.25	0.00
Rumba	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
Samba	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
Slowfox	0.00	0.00	0.00	0.33	0.00	0.00	0.33	0.00	0.00	0.33
Tango	0.00	0.00	0.00	0.25	0.00	0.00	0.25	0.50	0.00	0.00
V. Waltz	0.00	0.00	0.00	0.00	0.25	0.00	0.00	0.00	0.75	0.00
Waltz	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.00	0.50	0.00

Figure 5.4: Confusion matrix evaluated using *The Tutorials Model* in sliding window fashion on the whole videos from the *Dance Tutorials Dataset* validation set. We used sliding window of 300 frames with a stride of 60 frames (i.e. approx. 10 second window with 2 second stride).

5.3 Multi-modal classification

We performed an experiment combining classification based on video and classification based on audio - the two distinct modalities which can both be used in dance recognition.

As visual classifier, we use *The Tutorials Model*. For audio classification, we rely on the method introduced in [15]. We evaluate both of the classifiers independently on the *Dance Tutorials Dataset* validation set in sliding window manner - same as in Section 5.2.

Audio classifier setup. We use the model called 'densenet_ft' [15]. We use sliding windows corresponding to those used with the visual classifier- i.e.: 10 second window with stride of 2 seconds.

Audio classification results. Using the audio classifier with our choice of the sliding window parameters leads to Top-1 accuracy of 66.7% and Top-2 accuracy of 77.8%. This falls behind the accuracies reported in [15] but our results might not be comparable since in [15] a smaller window and a much finer stride were used.

In Figure 5.6a, we provide the confusion matrix for the audio classifier evaluated on *Dance Tutorials Dataset*.

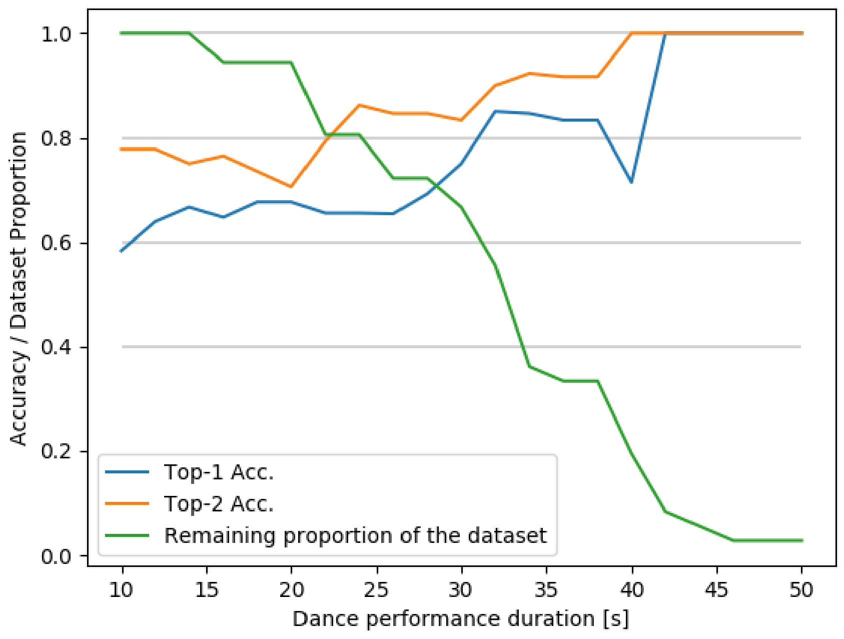


Figure 5.5: Dependency of classification accuracy on duration of classified sample. Note that for longer samples the support in the data declines as the duration of the classified samples exceeds durations of the dataset instances.

Modality:	**Audio**	**Video**	**Audio + Video**
Top-1 Acc.	66.7 %	72.2 %	83.3 %
Top-2 Acc.	77.8 %	86.1 %	97.2 %

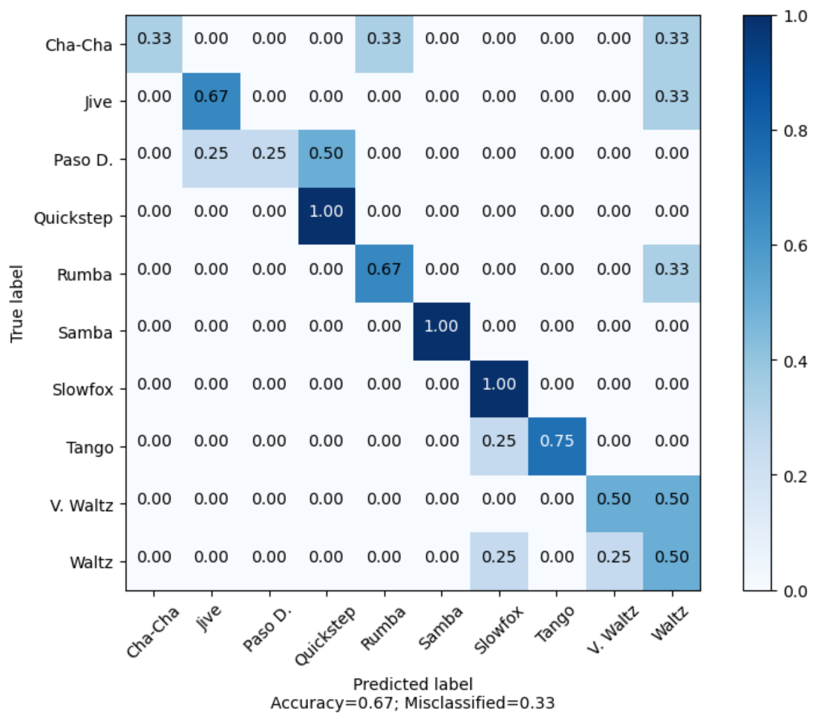
Table 5.7: Comparison of the classification accuracy using 2 modalities and their combination. Utilizing the sliding window approach with a 10 second window and a 2 second stride, the evaluation was performed on the *Dance Tutorials Dataset* validation set.

Audio-Visual classifier setup. We combine the two classifiers by summing the softmax scores returned for each of the windows. Using these combined scores we aggregate the decision over the whole instance and report the prediction.

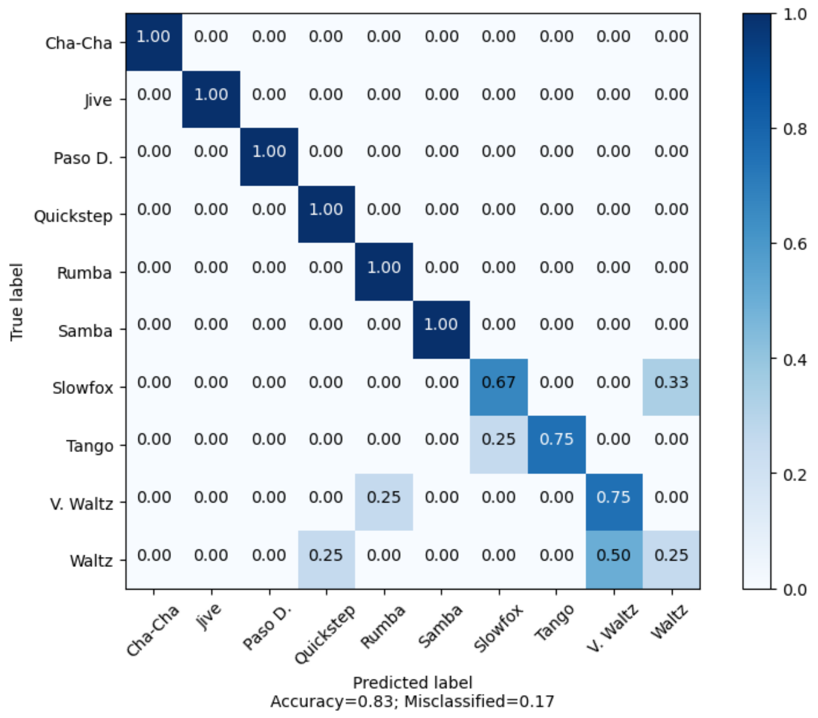
Audio-Visual classifier results. In Table 5.7, we provide the comparison of the accuracies achieved by classifiers based on different modalities. We see that both of the modalities are useful and can complement each other, as their combination significantly outperforms the classifiers based on a single modality only.

Furthermore, the complementarity is witnessed also by the confusion matrices - compare Figures 5.4,5.6a and 5.6b.

In Figure 5.7, we plotted dependency of accuracy on duration of classified samples for both of the modalities and their combination. We observe that



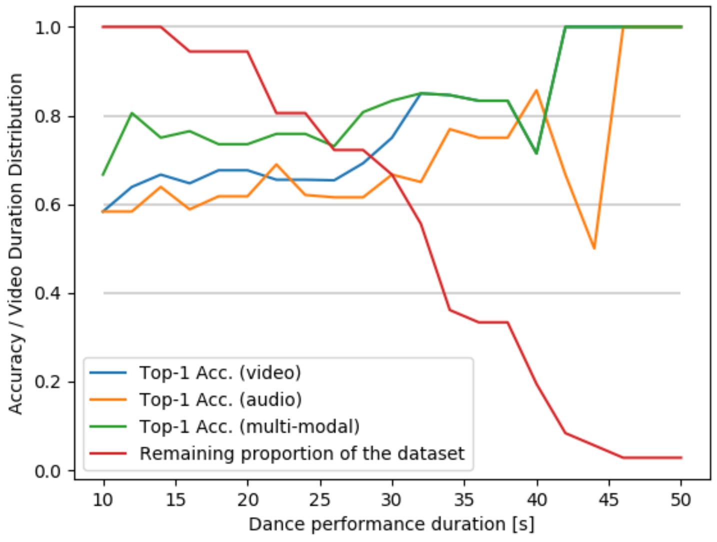
(a) : Confusion matrix for the audio-based classification



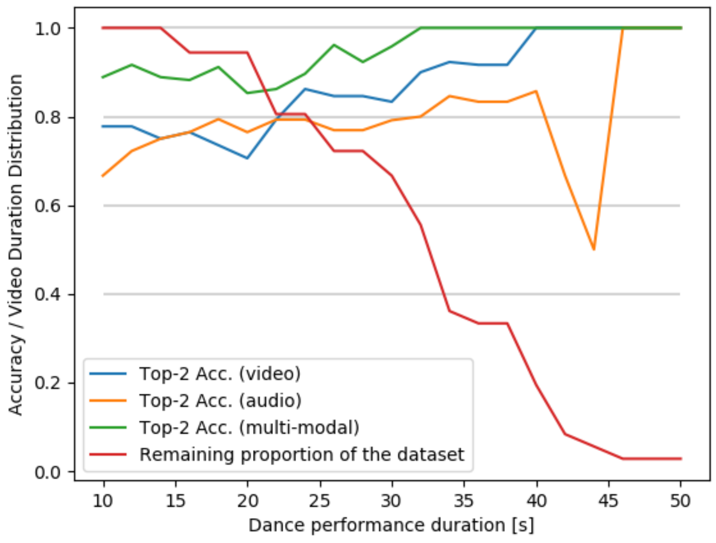
(b) : Confusion matrix for the combined audio-video-based classification

Figure 5.6: Confusion matrices evaluated using the audio and the combined audio-visual classifiers on the full videos from the validation set of *Dance Tutorials Dataset*. The evaluations were performed using a sliding window of 10 seconds with a stride of 2 seconds.

the improvement in Top-1 accuracy, brought by combining classification of both of the modalities, does not change too much during the initial 30 seconds of classification. Later the combined classifier followed the 'opinion' of the visual-based classifier. However, the longer sample durations are too rare in our dataset to provide enough support for drawing conclusions.



(a) : Top-1 Accuracies



(b) : Top-2 Accuracies

Figure 5.7: Accuracy of models using different modalities and their dependency on video duration.The evaluation was performed using a sliding window of 10 seconds with a stride of 2 seconds.

Chapter 6

Conclusion

In this thesis, we propose a method for visual-based classification of ballroom dances belonging to the so-called International Style[1]- 'The 10 Dances'.

One of our main contributions is the creation of 3 different datasets covering The 10 Dances. We make all of the three datasets publicly available [49].

1. *Dance Tutorials Dataset* is a novel dataset consisting of dance videos with a single clearly visible dance couple without other people on the background. See Section 2.2.1 for details.

2. *10-Let's Dance* is a dataset built by using 9 of the classes of an existing *Let's Dance* [13] dataset and complementing them by a novel collection of Viennese Waltz videos. The dataset mostly consists of videos with dances performed in crowded scenes. Details are provided in Section 2.2.2.

3. *YT8M Ballroom* is an experimental dataset based on preselected videos from the existing *Youtube-8M* [2] dataset. To annotate these videos by labels corresponding to The 10 Dances we proposed and performed an automated labeling procedure based on the audio classifier introduced in [15], see Section 2.2.3.

As a core of the approach towards the recognition of The 10 Dances we use the human pose estimation framework OpenPose [12].

Using the per-frame estimated human poses ('skeletons'), we create skeleton sequences by utilizing a tracking procedure which we developed.

Furthermore, we propose three different methods of normalization of the skeleton sequences, see Section 4.2.3. In an experiment however, we found out that none of the proposed normalization procedures helps us to train a more accurate model, see comparison in Table 5.2.

On top of the skeleton sequences we train a Graph Convolutional Network called MS-G3D [29] to classify the performed dances. We utilize a particular MS-G3D model which was pretrained on the *Kinetics-Skeleton* [11] dataset.

The best performing model trained on *Dance Tutorials Dataset* achieves accuracy of **64.8 % (Top-1)** and **76.0 % (Top-2)** when evaluated on non-overlapping 300 frames long video sequences from the validation set.

[1]See Table 1.1

When employing sliding window evaluation with the stride of 60 frames, the accuracy improves to **72.2 % (Top-1)** and **86.1 % (Top-2)** respectively. We conclude that the recognition tends to be more successful on longer samples, as is also witnessed by Figure 5.5.

To further improve the accuracy of the method, we suggest two potential improvements.

Firstly, there is a potential for improvement of the pose estimation stage. Nowadays a more accurate frameworks than OpenPose are available and a recent study [26] shows that using a more accurate pose estimation method significantly boosts the performance of skeleton-based action recognition models. Moreover, with a more advanced pose estimation framework the tracking could be performed implicitly during the pose estimation stage.

Secondly, we suggest replacing the fully connected classification layer of the model by a more sophisticated classifier - such as using more FC layers separated by a Dropout [54] layer.

The training of the model on *Dance Tutorials Dataset* with the training set enhanced by videos from *YT8M Ballroom.* does not significantly outperform the original model, compare Tables 5.4 and 5.2. We thus conclude that the videos in *YT8M Ballroom* do not depict the dance couples clearly enough so that the classifier could learn from them.

For future work we suggest to improve the selection of relevant segment for *YT8M Ballroom*, as the segment localization using the audio only does not guarantee there is a visible dance performance in the chosen segment. To this end, a person detector trained to recognize dancers could be employed with RGB frames of the videos.

A separate model trained on *10-Let's Dance* achieves accuracy of 55.5 % (Top-1) and 73.6 % (Top-2) respectively - with the evaluation performed on 300 frames long segments. We can see that the model is able to learn even from more challenging scenes, although the performance is inferior to the model trained and evaluated using *Dance Tutorials Dataset*.

Furthermore, we use the *10-Let's Dance* validation set to test how well the model trained on *Dance Tutorials Dataset* generalizes to new scenes. We achieve a Top-1 accuracy of 18.1 % only. We suspect that the model trained on *Dance Tutorials Dataset* is mostly accustomed to classifying well estimated skeletons, with most of the keypoints being detected correctly. In *10-Let's Dance* the dancers are often not contained completely in the picture, e.g. they might be filmed so that the feet are not visible etc.. Moreover, the crowded scenes produce occluded skeletons resulting in only partially detected skeletons as well.

We expected a better generalization from the model trained with the extra *YT8M Ballroom* training data, however the Top-1 accuracy improves only modestly - to 22.0 %. To help the model generalize better we suggest for future work to train it using data augmentation. For example the training set could be extended using skeleton sequences with randomly cropped skeletons.

Lastly, we test the multi-modal classification of The 10 Dances. We combine the visual model trained on *Dance Tutorials Dataset* with the audio classifier from [15]. We test both of the classifiers on the whole instances of the *Dance Tutorials Dataset* validation set and we reach an accuracy of **83.3 % (Top-1)** and **97.2% (Top-2)** respectively. This means a significant improvement in accuracy for both of the individual classifiers, see Table 5.7. We observe that the visual-based classifier performs better on the latin dances, see Figure 5.4, while the audio-based classifier performs better on Quickstep, Slowfox, Tango and Waltz - see Figure 5.6a. This complementarity makes the combined audio-visual classification very useful in the dance recognition domain.

For future work regarding the multi-modal classification, we suggest to try to improve the accuracy of the audio classifier by using a smaller window and stride, as the values we obtained fall behind the claims of its authors [15]. Furthermore, a more profound way of combining the classifiers could be examined in future. One of the ways could be training a SVM classifier on top of the softmax scores returned by both of the networks, as suggested in [17].

Appendix A
Supplementary tables and figures

Norm.:		**No trans.**		**No fr. norm**		**No norm**		**Full norm**	
Training depth	Base l.r.	Best ep.	Top1 Acc.	Best ep.	Top1 Acc.	Best ep.	Top1 Acc.	Best ep.	Top1 Acc.
Depth 1	0.005	42	61.6	35	51.2	19	60.8	55	52
Depth 1	0.01	20	63.2	36	56.8	50	64.8	19	56
Depth 1	0.05	24	62.4	25	58.4	60	61.6	25	59.2
Depth 1	0.1	21	61.6	58	60.0	23	56.0	46	54.4
Depth 2	0.005	11	61.6	21	55.2	51	57.6	34	59.2
Depth 2	0.01	43	61.6	63	54.4	29	64.8	21	56.0
Depth 2	0.05	23	60.8	27	57.6	61	60.0	43	60.0
Depth 2	0.1	12	60.8	41	58.4	22	60.0	32	57.6
Depth 3	0.005	14	62.4	56	58.4	31	56.8	34	57.6
Depth 3	0.01	9	64.8	51	55.2	19	58.4	9	57.6
Depth 3	0.05	23	64.0	18	59.2	12	56.0	62	58.4
Depth 3	0.1	34	60.0	36	59.2	31	57.6	60	53.6
Depth 4	0.001	64	57.6	57	55.2	30	59.2	29	56.0
Depth 4	0.005	8	62.4	50	60.0	33	59.2	42	60.8
Depth 4	0.01	10	60.0	20	59.2	8	61.6	47	58.4
Depth 4	0.05	9	60.8	37	60.0	55	58.4	35	58.4
Depth 5	0.001	11	58.4	21	57.6	28	55.2	48	55.2
Depth 5	0.005	41	60.8	36	58.4	15	57.6	55	59.2
Depth 5	0.01	51	60.0	40	57.6	31	56.0	22	57.6
Depth 5	0.05	31	62.4	57	56.0	26	63.2	34	60.8

Table A.1: Overview of the trainings performed in order to systematically evaluate the contribution of normalizing the skeleton sequence using the normalization variants suggested in Section 4.2.3.

Training depth	Depth 1				Depth 2			
Base learn. rate	0.005	0.01	0.05	0.1	0.005	0.01	0.05	0.1
Best epoch	48	3	26	44	17	7	89	42
Top-1 Acc. [%]	56.8	**58.4**	58.4	55.2	56.8	56.8	57.6	32.8

Table A.2: Summary of the trainings on *Dance Tutorials Dataset* using the bone-stream network. The trainings were run for 100 epochs, using SGD optimizer and a weight decay of 0.0003.

```
{
    "data":[
        {
            "frame_index": int,
            "skeleton": [
                {
                "pose": [
                        $x_0$,
                        $y_0$,
                        $x_1$,
                        ...,
                        $x_{17}$,
                        $y_{17}$
                    ],
                "score": [
                        $c_0$,
                        ...,
                        $c_{17}$
                    ]
                },
                ...
            ]
        },
        ...
    ],
    "label": str,
    "label_index": int
}
```

Figure A.1: JSON Template for representing the output of the skeleton pre-processing. It can be transformed into a numpy array input of the network using the script provided by the authors of [29]. One such JSON represents up to 300 frames of a video, each entry of the "data" list corresponds to a single frame, each entry of the "skeleton" list corresponds to a single estimated pose within that frame. The list "pose" contains the coordinates of the keypoints and "score" contains their confidences. The "label" is one of The 10 Dances with "label_index" being an integer from 0 to 9 uniquely assigned to each label.

Appendix B
Acronyms

API Application Programming Interface. 24

CNN Convolutional Neural Network. 15, 17, 19

COCO Common objects in context, dataset name and a name for a skeleton format. 24, 25

ConvNet Convolutional Network. 16, 17

CTU Czech Technical University in Prague. 24

DNN Deep Neural Network. 15

FC Fully Connected (layer). viii, 36, 52

GCN Graph Convolutional Network. vi, 18–20, 27, 37

GPU Graphics Processing Unit. 23, 24, 40

HMDB-51 Human Motion Database 51, human action recognition dataset. 16

ID Identification, typically used as unique code for each Youtube video. 8–10, 13

LSTM Long-Short-Term-Memory a type of a neural network. 16, 17

MS-G3D The name of a particular GCN. iv–vi, 20, 21, 27, 28, 34–37, 40, 51

NTU RGB+D human action dataset obtained using depth sensor, represented by 3D skeletons. 17

RCI Research Center for Informatics. 24

RGB Red Green Blue, a color space for representation of a digital image. 7, 9, 10, 15–17, 21, 52

RGB-D Red Green Blue + Depth. 21

SGD Stochastic Gradient Descent. viii, 40, 43, 56

SLURM Simple Linux Utility for Resource, a workload manager. 24

ST-GCN Spatial-Temporal Graph convolutional Network. 19, 20

STAF Spatial-Temporal Affinity Fields. 25

TCN Temporal Convolutional Network. vi, 37

UCF-101 name of a human action recognition dataset. 16

URL Address of a web page. 5

w.r.t. with respect to. 32, 33

Appendix C
Bibliography

[1] Pareek, P., Thakkar, A. A survey on video-based Human Action Recognition: recent updates, datasets, challenges, and applications. *Artif Intell Rev 54, 2259–2322* (2021). https://doi.org/10.1007/s10462-020-09904-8

[2] Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B. & Vijayanarasimhan, S. *Youtube-8m: A large-scale video classification benchmark. ArXiv Preprint ArXiv:1609.08675.* (2016)

[3] Pfeiffenberger P., Gaw S. Google I/O 2013 - semantic video annotations in the Youtube Topics API: Theory and applications. https://www.youtube.com/watch?v=wf_77z1H-vQ

[4] Google Research: http://research.google.com/youtube8m/ Accessed: 7/22/2021.

[5] Carreira, J. & Zisserman, A. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. *2017 IEEE Conference On Computer Vision And Pattern Recognition (CVPR).* pp. 4724-4733 (2017)

[6] Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M. & Zisserman, A. The Kinetics Human Action Video Dataset. (2017)

[7] Deng, J., Dong, W., Socher, R., Li, L., Li, K. & Fei-Fei, L. ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference On Computer Vision And Pattern Recognition.* pp. 248-255 (2009)

[8] Carreira, J., Noland, E., Banki-Horvath, A., Hillier, C. & Zisserman, A. A Short Note about Kinetics-600. (2018)

[9] Carreira, J., Noland, E., Hillier, C. & Zisserman, A. A Short Note on the Kinetics-700 Human Action Dataset. (2019)

[10] Smaira, L., Carreira, J., Noland, E., Clancy, E., Wu, A. & Zisserman, A. A Short Note on the Kinetics-700-2020 Human Action Dataset. (2020)

[11] Yan, S., Xiong, Y. & Lin, D. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. *AAAI.* (2018)

[12] Cao, Z., Simon, T., Wei, S. & Sheikh, Y. Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition (CVPR)*. (2017,7)

[13] Daniel Castro & Essa, I. Let's Dance: Learning From Online Dance Videos. *Eprint ArXiv:2139179*. (2018)

[14] Güler, R., Neverova, N. & Kokkinos, I. DensePose: Dense Human Pose Estimation In The Wild. (2018)

[15] Pavlín, T., Čech, J. & Matas, J. Ballroom Dance Recognition from Audio Recordings. *2020 25th International Conference On Pattern Recognition (ICPR)*. pp. 2142-2149 (2021)

[16] Shannon, C. A mathematical theory of communication.. *Bell Syst. Tech. J.*. **27**, 379-423 (1948), `http://dblp.uni-trier.de/db/journals/bstj/bstj27.html#Shannon48`

[17] Wysoczańska., M. & Trzciński., T. Multimodal Dance Recognition. *Proceedings Of The 15th International Joint Conference On Computer Vision, Imaging And Computer Graphics Theory And Applications - Volume 5: VISAPP,*. pp. 558-565 (2020)

[18] Carreira, J. & Zisserman, A. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. *2017 IEEE Conference On Computer Vision And Pattern Recognition (CVPR)*. pp. 4724-4733 (2017)

[19] (Facebook AI), S. Skeleton Based Action Recognition On NTU RGB+D. , `https://paperswithcode.com/sota/skeleton-based-action-recognition-on-ntu-rgbd`

[20] (Facebook AI), S. Skeleton Based Action Recognition On NTU RGB+D 120. , `https://paperswithcode.com/sota/skeleton-based-action-recognition-on-ntu-rgbd-1`

[21] Shahroudy, A., Liu, J., Ng, T. & Wang, G. NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis. (2016)

[22] Liu, J., Shahroudy, A., Perez, M., Wang, G., Duan, L. & Kot, A. NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding. *IEEE Transactions On Pattern Analysis And Machine Intelligence*. **42**, 2684-2701 (2020,10), http://dx.doi.org/10.1109/TPAMI.2019.2916873

[23] (Facebook AI), Skeleton Based Action Recognition Benchmarks. , `https://paperswithcode.com/task/skeleton-based-action-recognition`

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

[25] Facebook AI, Pose estimation on MPII Human Pose. , `https://paperswithcode.com/sota/pose-estimation-on-mpii-human-pose`

[26] Duan, H., Zhao, Y., Chen, K., Shao, D., Lin, D. & Dai, B. Revisiting Skeleton-based Action Recognition. (2021)

[27] Shi, L., Zhang, Y., Cheng, J. & Lu, H. Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition. *CVPR*. (2019)

[28] Obinata, Y. & Yamamoto, T. Temporal Extension Module for Skeleton-Based Action Recognition. (2020)

[29] Liu, Z., Zhang, H., Chen, Z., Wang, Z. & Ouyang, W. Disentangling and Unifying Graph Convolutions for Skeleton-Based Action Recognition. *Proceedings Of The IEEE/CVF Conference On Computer Vision And Pattern Recognition*. pp. 143-152 (2020)

[30] Zheng, C., Wu, W., Yang, T., Zhu, S., Chen, C., Liu, R., Shen, J., Kehtarnavaz, N. & Shah, M. Deep Learning-Based Human Pose Estimation: A Survey. (2020)

[31] Toshev, A. & Szegedy, C. DeepPose: Human Pose Estimation via Deep Neural Networks. *2014 IEEE Conference On Computer Vision And Pattern Recognition*. pp. 1653-1660 (2014)

[32] Krizhevsky, A., Sutskever, I. & Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*. **25** (2012), https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[33] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. Rethinking the Inception Architecture for Computer Vision. *2016 IEEE Conference On Computer Vision And Pattern Recognition (CVPR)*. pp. 2818-2826 (2016)

[34] Pishchulin, L., Insafutdinov, E., Tang, S., Andres, B., Andriluka, M., Gehler, P. & Schiele, B. DeepCut: Joint Subset Partition and Labeling for Multi Person Pose Estimation. *IEEE Conference On Computer Vision And Pattern Recognition (CVPR)*. (2016), http://arxiv.org/abs/1511.06645

[35] Peter Naftaliev, Overview of Human Pose Estimation Neural Networks Accessed: 7/28/2021 , `https://towardsdatascience.com/overview-of-human-pose-estimation-neural-networks-hrnet-higherhrnet-architectures-and-faq-1954b2f8b249`

[36] Cao, Z., Simon, T., Wei, S. & Sheikh, Y. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *CVPR*. (2017)

[37] Ginés Hidalgo, Zhe Cao, Tomas Simon, Shih-En Wei, Yaadhav Raaj, Hanbyul Joo, and Yaser Sheikh, OpenPose. , Accessed: 8/8/2021 https://github.com/CMU-Perceptual-Computing-Lab/openpose

[38] Newell, A., Huang, Z. & Deng, J. Associative Embedding: End-to-End Learning for Joint Detection and Grouping. *Advances In Neural Information Processing Systems*. **30** (2017), https://proceedings.neurips.cc/paper/2017/file/8edd72158ccd2a879f79cb2538568fdc-Paper.pdf

[39] tugstugi, dl-colab-notebooks Accessed: 14/4/2020 , https://github.com/tugstugi/dl-colab-notebooks

[40] Fang, H., Xie, S., Tai, Y. & Lu, C. RMPE: Regional Multi-person Pose Estimation. *ICCV*. (2017)

[41] Sun, K., Xiao, B., Liu, D. & Wang, J. Deep High-Resolution Representation Learning for Human Pose Estimation. *CVPR*. (2019)

[42] Raaj, Y., Idrees, H., Hidalgo, G. & Sheikh, Y. Efficient Online Multi-Person 2D Pose Tracking With Recurrent Spatio-Temporal Affinity Fields. (2019,6)

[43] Raaj, Y., Idrees, H., Hidalgo, G. & Sheikh, Y., STAF Algorithm https://github.com/soulslicer/STAF/tree/staf

[44] Gssrao Youtube-8m Videos and Frames Generator. , Accessed: 7/20/2021 https://github.com/gsssrao/youtube-8m-videos-frames

[45] Bradski, G. The OpenCV Library. *Dr. Dobb's Journal Of Software Tools*. (2000)

[46] Harris, C., Millman, K., Walt, S., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M., Brett, M., Haldane, A., Río, J., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. & Oliphant, T. Array programming with NumPy. *Nature*. **585** pp. 357-362 (2020)

[47] Bozinovski, S. Reminder of the First Paper on Transfer Learning in Neural Networks, 1976. *Informatica (Slovenia)*. **44** (2020)

[48] Francois Chollet, Transfer learning & fine-tuning. Accessed:8/7/2021,https://keras.io/guides/transfer_learning/

[49] Kouba, P. BallroomDanceDatasets. Accessed: 8/7/2021, https://github.com/KoubaPetr/BallroomDanceDatasets/

[50] Kouba, P. PoseEstimationExamples. Accessed: 8/7/2021 https://github.com/KoubaPetr/PoseEstimationExamples

[51] Sijie Zan, st-gcn, Accessed: 8/7/2021 , https://github.com/yysijie/st-gcn/blob/master/OLD_README.md

[52] Ziyu Liu, MS-G3D, Accessed: 8/9/2021 , `https://github.com/kenziyuliu/MS-G3D`

[53] Elisha Odemakinde, Human Pose Estimation with Deep Learning – Ultimate Overview in 2021 Accessed: 8/8/2021 , `https://viso.ai/deep-learning/pose-estimation-ultimate-overview/`

[54] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*. **15**, 1929-1958 (2014,1)

[55] Kuehne, H., Jhuang, H., Garrote, E., Poggio, T. & Serre, T. HMDB: A large video database for human motion recognition. *2011 International Conference On Computer Vision*. pp. 2556-2563 (2011)

[56] Soomro, K., Zamir, A. & Shah, M. UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild. *CoRR*. **abs/1212.0402** (2012), `http://dblp.uni-trier.de/db/journals/corr/corr1212.html#abs-1212-0402`