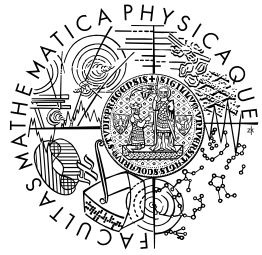

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Tomáš Pavlín

Dance Recognition from Audio Recordings

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis: Ing. Jan Čech, Ph.D.
Study programme: Computer Science
Study branch: IUI

Prague 2020

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Firstly, I would like to thank to my supervisor Jan Čech for his valuable advises, guidance and remarkable effort. I am grateful for his helpful comments, encouragement and assistance he has provided. Further, I would like to thank to prof. Jiří Matas for his enthusiasm about the thesis topic and for his inspiring ideas. I also thank to Jan Hajič jr. for useful insights and comments regarding music topics.

Title: Dance Recognition from Audio Recordings

Author: Tomáš Pavlín

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Ing. Jan Čech, Ph.D., Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Cybernetics

Abstract: We propose a CNN-based approach to classify ten genres of ballroom dances given audio recordings, five latin and five standard, namely Cha Cha Cha, Jive, Paso Doble, Rumba, Samba, Quickstep, Slow Foxtrot, Slow Waltz, Tango and Viennese Waltz. We utilize a spectrogram of an audio signal and we treat it as an image that is an input of the CNN. The classification is performed independently by 5-seconds spectrogram segments in sliding window fashion and the results are then aggregated. The method was tested on following datasets: Publicly available Extended Ballroom dataset collected by Marchand and Peeters, 2016 and two YouTube datasets collected by us, one in studio quality and the other, more challenging, recorded on mobile phones. The method achieved accuracy 93.9%, 96.7% and 89.8% respectively. The method runs in real-time. We implemented a web application to demonstrate the proposed method.

Keywords: ballroom dance genre classification CNN audio music

Contents

1. Introduction

This study presents a method for dance genre classification by convolutional neural networks (CNN). Dance genre classification is a topic within the research field music information retrieval (MIR) and it aims to recognize a dance class (Waltz, Tango Jive, ...) from a given audio recording.

The problem of predicting dance class from audio recordings is interesting from both theoretical and practical perspectives. The core of the dance recognition problem is to encode low-level information from a raw audio signal and to produce a semantic high-level information. The process of extracting such high-level information from music is similar to music perception by humans, and can be used for applications as music discovery and recommendation. We benefit from employing the dance music, because a large amount of audio data is publicly available. Various audio recordings of dance music can be extracted either from YouTube or from video recordings of dance competitions, that experience increasing popularity. The categorization of such music data is usually within video or track titles, thus the data can be easily labeled. Moreover, dance recognition is beneficial for beginners or amateur dancers that have difficulties to recognize the dance genre from the music. To the best of our knowledge, there is no such commercial application that would help the dancers to recognize a dance from music.

Rather than a dance genre classification, many researches focused on a related problem, music genre recognition, for the past decades. The problem of predicting a dance class from audio recordings is largely unexplored.

A brief explanation of our technique for dance genre classification will follow. First, we convert given audio recording to image representation called spectrogram. Spectrogram is temporal representation of spectrum of frequencies. The spectrogram is cut to segments in sliding window fashion, each segment corresponding to audio of given length. A convolutional neural network (CNN) takes each spectrogram segment as an input and outputs the classification results. CNNs are deep convolutional networks that only consists of convolutional layers and they do not rely on any fully connected layer. Our model is trained using a set of 4000+ audio recordings of ballroom music, to predict a correct dance style. We achieved accurate results on both, our test set that we publicly provide for research community, and on the novel Extended Ballroom dataset [Marchand and Peeters, 2016a].

Our method might me useful for music information retrieval (MIR), because it propose novel approach of extracting high-level features of audio signal, i.e. the dance classes. Our classifier is fast and classify dance music in real-time with only a small latency. Moreover, we demonstrate our approach on a working web application. We believe that the method has potential as a commercial application meant for the beginner dancers.

The contributions of the thesis are: (1) We proposed a novel CNN based method for dance recognition, (2) we collected the test dataset that we made publicly available for research community, and (3) we implemented the web application as a demonstration.

The remainder of this study is organized as follows. We present overview of

related work in Section 2 and explain our method in Section 3. The method is quantitatively evaluated by various experiments in Section 4. In the experiment section, we firstly describe our datasets for training, testing and validation in Section 4.2. To compare with our results, we describe baseline algorithm in Section 4.3, followed by presenting results of our method in Section 4.4. Several CNN architectures are used for dance recognition and compared with each other in Section 4.5. Section 4.6 provides results of our method on different datasets with ballroom dances including dataset with low quality data. Unsatisfactory results on the low quality dataset motivate us to adjust our model to handle the low quality data in Section 4.7. Several classification results for notable audio recordings are shown in Section 4.8 as well as working web application is demonstrated. Finally, conclusion is presented in Section 5.

2. Related Work

Dance recognition from audio samples relates to interdisciplinary area called Music Information Retrieval (MIR) [Downie, 2003]. The aim of this science is to analyse and to describe musical data and to retrieve information concerning music files.

Significant research has been made recently in the field with different objectives. Music genre classification task is common in this area, although there is no significant focus on dance music recognition.

Among the small number of studies regarding dance music recognition, we highlight three. Dixon et al. [2003] classify dance music by timing information as tempo and meter (periodicity). The authors use the timing information to describe each dance class by functional language rules. The approach was further improved by Chew et al. [2005] by relying not only on periodicity patterns in the music, but also on the accent patterns.

Lastly, Marchand and Peeters [2016b] present scale and shift-invariant time-frequency representation of audio content. The authors propose a classifier of ballroom dance music with promising results, although there is no significant focus on the classifier and the testing protocol is unclear.

We also explored mobile phone applications related to dance music and we can highlight DancePicker[1]. The authors present that the application can find the right dance for given music. However, the music is supposed to be specified by title and artist name and the music database is limited. We did not find any application with functionality of detecting dance from audio recordings.

In conclusion, to the best of our knowledge, there is neither commercial application nor a recent paper on dance recognition. Music classification studies does not focus on dance recognition, but rather on music genres classification (MGC) with the goal to predict music genre (e.g. classical, electronic, jazz, rock, metal) from given audio recording.

2.1 Music Genres Classification

Bellow we review stat-of-the-art literature of the related problem, MGC, and the methodology we are inspired by.

Musical genres are classes in which the audio samples share similar audio features as instruments, tempo, chords and rhythmic patters [Costa et al., 2012]. However, the definition of the classes may be vague. The reason is that to distinguishing musical genres and information about them may be challenging task even for humans [Lippens et al., 2004].

There are several approaches for music genres classification. Many papers attempted to solve the task by extracted handcrafted features from audio, such as Mel Frequency Cepstral Coefficients [Logan et al., 2000]. These features routinely used in speech recognition are used as input to machine learning classifier such as Support Vector Machine (Tzanetakis and Cook [2002]; Nakai et al. [2018]). Novel architectures use image-like representation of the audio, called spectrogram, along

[1] `http://dancepicker.com`

with convolution neural networks. The lastly mentioned is the state-of-the-art approach [Bahuleyan, 2018].

2.2 Classification using Neural Networks

Modern approaches use neural networks to solve the problem of music classification. Most of these approaches rely on spectrogram and convolutional neural networks (CNN) (Dieleman and Schrauwen [2014]; Pons et al. [2016]; Choi et al. [2016]; Pons et al. [2017]). Further Dieleman and Schrauwen [2014] and Pons et al. [2017] compare the mentioned method with approach classifying the music using raw audio without constructing the spectrogram. The authors achieved good results, however they do not outperform the spectrogram-based approaches.

Several CNN-based approaches employ neural networks, that are pre-trained for different task, e.g. image categorization, with larger dataset. These architectures take advantage of transfer learning. While weights in the convolution layers are kept fixed, the weights in the classification part of neural network are trained in order to predict the correct music genre (Image 2.1). The idea behind is that the convolutional blocks provide invariant representation, while the last layers perform no-linear classification.

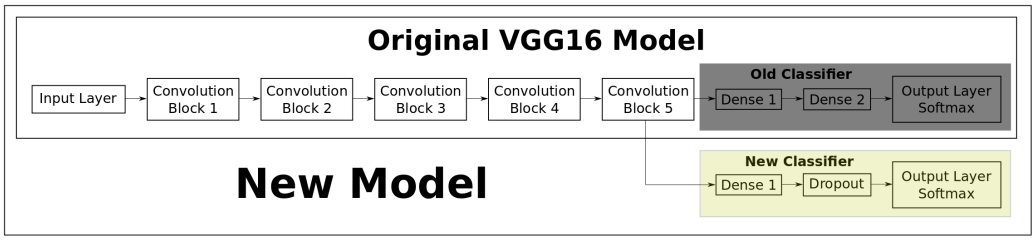

Figure 2.1: Transfer learning architecture on convolutional neural network (Source: `https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/10_Fine-Tuning.ipynb`)

Among the studies that use a pre-trained CNN, we can highlight Bahuleyan [2018]. The authors employed VGG16, CNN network which was the top performing model in the ImageNet Challenge 2014 [Simonyan and Zisserman, 2014]. The network consists of 5 convolution blocks followed by densely connected layers predicting class of a given image. Bahuleyan [2018] used pre-trained convolution layers of VGG-16 with spectrogram as an input. The pre-trained layers were followed by densely connected layers to predict genre of the music.

We also highlight Tang et al. [2018]. The authors employed hierarchical Long Short Term Memory (LSTM), a recurrent neural network, to recognise music genres.

Further, Oramas et al. [2018] studied multimodal music genre classification with use of deep neural networks. Besides audio, the authors used images of cover photos and text of reviews.

2.3 Datasets

The method for music classification described above requires training dataset with music files. Next, we will list significant datasets used for dance genre classification and related tasks.

2.3.1 Datasets with Dance Genres

In this section, we describe two datasets that contain dance music and can be used for dance genre classification task. The first is the **Ballroom Dataset** introduced for a conference ISMIR 2004 (5th International Conference on Music Information Retrieval). The authors used website *www.ballroomdancers.com* that sells audio CDs of ballroom dances and offers 30 seconds preview of each track to listen for free. It contains 698 music recordings divided into 8 genres that represent ballroom dances (Cha Cha Cha, Jive, Quickstep, Rumba, Samba, Tango, Viennese Waltz, Waltz). The disadvantage of this dataset is low audio quality and small amount of both audio recordings and dance classes.

Other dataset with dance music was made by improving the Ballroom Dataset and is referred as **Extended Ballroom Dataset** [Marchand and Peeters, 2016a]. Similarly to the Ballroom Dataset, it consists of 30 seconds recordings obtained from the website. However, compared to the Ballroom Dataset, the Extended Ballroom Dataset contains 6 times more recordings (4180) and it has better audio quality. Further, besides the classes of the Ballroom Dataset, the Extended Ballroom Dataset contains 5 new dance classes as Foxtrot, Paso Doble, Salsa, Slow Waltz and West Coast Swing. For every recording, the dataset also provides annotations as tempo, artist, song title and album name.

2.3.2 Datasets with Music Genres

For the reason, that music genre classification task is more common than dance genre classification, there is higher amount datasets labeled with music genres. Two of them will be highlighted in this section.

GTZAN Genre Collection [Tzanetakis and Cook, 2002] has been extremely popular. The dataset contains 10 genres as classical, country, disco, hiphop, jazz, rock, blues, reggae, pop and metal. Every genre is represented by 100 tracks each 30 seconds long.

Other interesting approach to obtain recordings of musical genres was used by Bahuleyan [2018]. Dataset with 2.1 million sound clips extracted from YouTube videos was used [Gemmeke et al., 2017]. The dataset contains 10 second sound segments that are specified by YouTubeID of the corresponding videos along with the start end end times. The segments are divided into 632 audio classes and, while most of these classes relate to audio events and not to the music, Bahuleyan [2018] extracted 7 of the classes belonging to music genres. Such approach resulted into 40 540 audio recordings of pop, rock, hip hop, techno, rhythm blues, vocal and reggae.

2.3.3 Other Datasets

It should be noted, that there is also the **Million Song Dataset** [Bertin-Mahieux et al., 2011] that is popular in MIR. While it is made from 1 million songs, it does not contain audio but audio features and metadata only.

3. Method

Given an audio recording of a ballroom dance, our goal is to classify it into one of ten ballroom dance classes, five standard and five latin:

1. Cha Cha Cha
2. Jive
3. Paso Doble
4. Quickstep
5. Rumba
6. Samba
7. Slow Foxtrot
8. Slow Waltz
9. Tango
10. Viennese Waltz

The classes correspond to the International Style dances popular on dance competitions. We divide our approach into three stages. Firstly, we convert audio signal to image representation called spectrogram. The spectrogram is cut into short overlapping segments that are then classified independently. In the next step, we employ convolutional neural network to classify the spectrogram segments. Finally, we aggregate classification results from several segments to classify the whole recording. Next, we describe the three stages.

3.1 Converting Audio to Image Representation

Before we employ neural networks to classify the audio, we perform pre-processing by converting the raw audio signal to image representation called spectrogram. We have chosen this approach for the reason that it is currently state-of-the-art method in music genre classification [Bahuleyan, 2018].

3.1.1 Spectrogram

Spectrogram is a frequency-temporal 2D representation of audio signal having time on the horizontal axis and frequency on the vertical axis. It represents the spectrum (magnitudes of given frequencies) within a given time window as can be seen in Figure 3.1. Short-time Fourier transformation (STFT) is used for spectrogram computation.

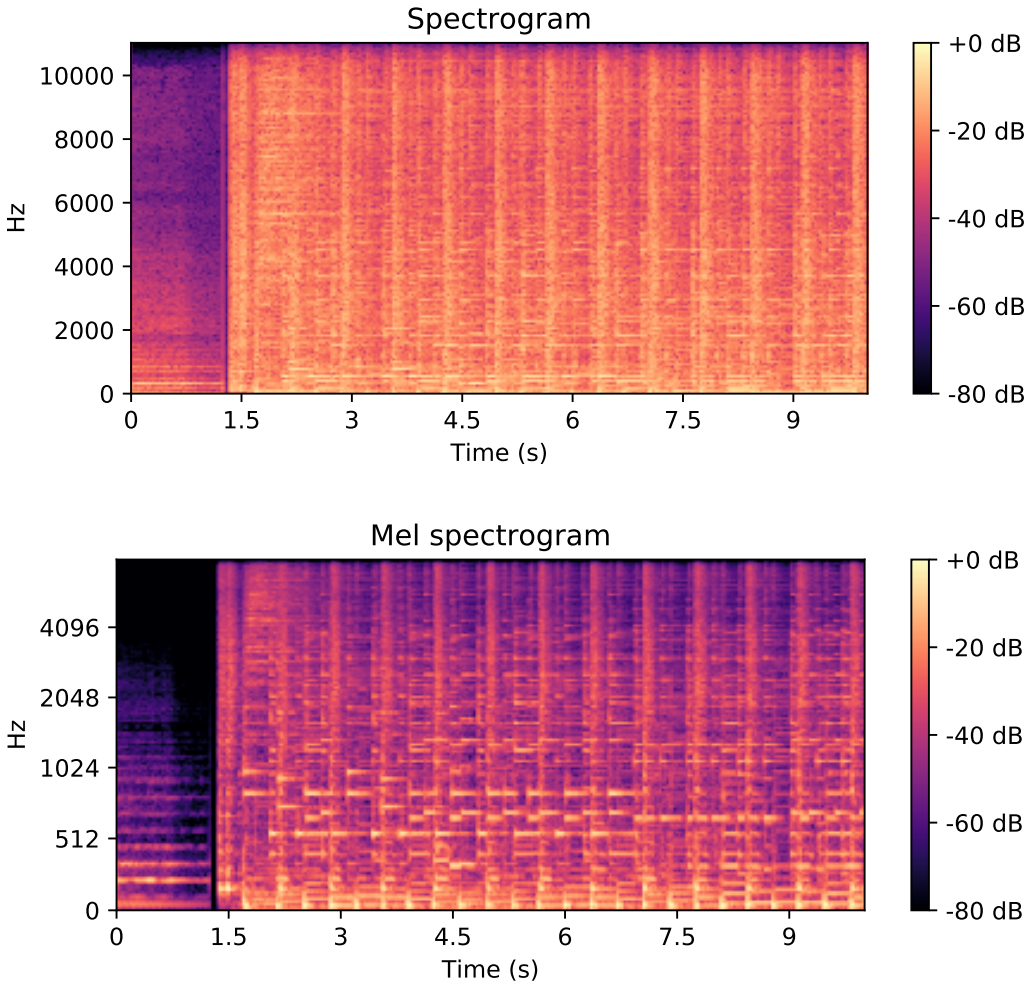

Figure 3.1: Comparison of normal-scaled and and mel-scaled spectrogram created from 10 second recording of Jive music. The 1.4 seconds in the beginning correspond to the duration before the song start.

3.1.2 Mel Spectrogram

In this study, we convert audio signal to MEL spectrogram that has the frequency axis scaled logarithmically. See comparison of these spectrograms in Figure 3.1.

Parameter $n\_mels = 224$ represents the height of the spectrogram. The width of the spectrogram depends on audio sample length. With used parameters, one spectrogram column corresponds with a time span of the following size:

$$\frac{hop\_length}{sr} = \frac{512}{22050} \doteq 0.023 = 23\,\text{ms}$$

In our research, we converted all audio files to image representation using MEL spectrograms. The 2D representation of the input allows us to use the same convolutional network architecture that has been used in computer vision to categorize the images.

3.2 Convolutional Neural Network

Next, we cut the spectrogram to segments of size 224×224 in sliding window fashion. It means the segment has original spectrogram height ($n\_mels = 224$) but lower width than the original spectrogram as depicted in Figure 3.2.

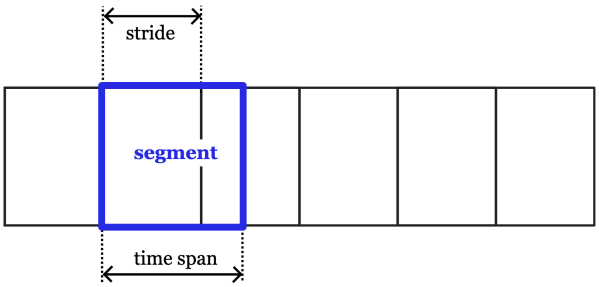

Figure 3.2: The spectrogram is cut to segments in sliding window fashion. The segment size is 224×224 and its width correspond to time span of 5.2 seconds.

The segment width corresponds to time span that is given by:

$$timespan = \frac{224 \cdot hop\_length}{sr} = \frac{224 \cdot 512}{22050} \doteq 5.20\,\mathrm{s}$$

The spectrogram segment with length of 5.20 seconds is further used for classification using convolutional neural network. Experiments show the used segment length is long enough to predict correct dance style accurately. However, section 3.3 describes mechanism to classify samples that are longer than segment length. The spectrogram segment of size 224×224 is used as input to our model.

3.2.1 CNN Architecture

While there is significant number of convolutional neural networks architectures, we use Dense Convolutional Network (DenseNet) introduced by Huang, Liu, Van Der Maaten, and Weinberger [2017].

Dense Convolutional Network

DenseNet is a recent convolutional network that outperforms state-of-the-art approaches such as ResNet [He et al., 2016] in various aspects. It requires less computational power to achieve high accuracy.

The architecture is inspired by residual connections in ResNet and embrace the observation of advantage of shorter connections between layers. While ResNet contains residual connections between consecutive layers, DenseNet connect each layer with every subsequent layer.

The main building block of DenseNet is dense block depicted in Figure 3.3. Dense block consists of convolutional layers where each layer obtains additional inputs from all preceding layers and passes its output to all subsequent layers. In contrast to ResNet, the features are not combined by summation before they are passed into a layer. Instead, the features are concatenated. Consequently, the layer l has l inputs corresponding to outputs to all preceding layers within a

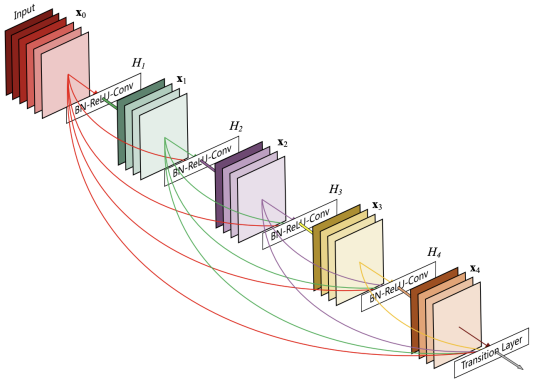

Figure 3.3: Dense block with 5 layers. Each layer takes all preceding feature-maps as input. (Reproduction from Huang et al. [2017])

given dense block. Similarly each layer output is passed to every $L-l$ subsequent layer.

Note, that the number of input channels increases with proceeding convolution layer, since each layer input consists of concatenated outputs of all preceding layers. The increased number of channels for each convolutional layer is constant and referred as *growth rate*. By design, the *growth rate* also corresponds to the number of output channels of each convolution layer.

Although the output feature-maps are narrow, the number of input channels increases substantially. Hence 1×1 convolutions are employed to reduce number of input channels before applying 3×3 convolutions.

The 1×1 convolutions are applied to the output of each dense block excluding the last one. Then, 2×2 average pooling layers are applied to reduce feature-map size. The output of the last pooling layer is followed by global average pooling. Finally, to predict the model output, the fully connected linear layer is applied along with softmax function (we will refer the last linear layer as classification layer). The architecture of the DenseNet is depicted in Figure 3.4.

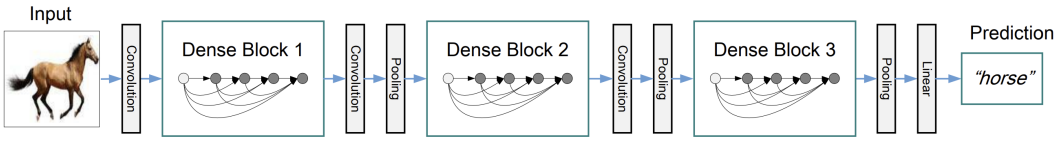

Figure 3.4: Architecture of DenseNet with three dense blocks. (Reproduction from Huang et al. [2017])

It should be noted, that first dense block is preceded by 7×7 convolution layer that has RGB image of size 224×224 as input. Moreover, dropout [Srivastava et al., 2014] is applied after each 3×3 convolution to prevent overfitting.

While DenseNet has typically either 3 or 4 dense blocks, the authors experimented with other various meta parameters. In our research, DenseNet 161 is used with following parameters:

- **Dense blocks sizes: 6, 12, 36, 24** - Number of convolutional layers in four dense blocks

- **Initial number of features: 96** - Output features after initial convolution.
- **Growth rate: 48**

In our study, pre-trained DenseNet is used with parameters learned on ImageNet [Deng et al., 2009]. The last classification layer is replaced with layer of 10 neuron output corresponding to the number of dance classes.

3.2.2 Training

This section describes, how we train the model. For each training iteration, we create a batch by the following approach.

Let $C = (c_1, \ldots, c_n)$ be the dance classes and let $D = ((x_1, y_1), \ldots, (x_m, y_m))$ be the labeled audio recordings where $x_i = (x_{i,1}, x_{i,2}, \ldots)$ corresponds to the audio signal and $y_i \in C$. We obtain the training sample by the following process.

First, we choose a class $c_i \in C$ randomly. Next, audio recording with its label $(x_j, y_j) \in D$ is chosen randomly from the dataset such that $y_j = c_i$. We select the audio recording sampling uniformly the dance classes to compensate class-imbalance in the training set.

Given the audio recording, we obtain corresponding MEL spectrogram $S_j = S(x_j)$ where S is a function that converts audio signal to the spectrogram. Spectrogram S_j can be represented by a matrix of size $224 \times d_j$ and we suppose only samples where $d_j \geq 224$:

$$S_j = \begin{bmatrix} s_{1,1} & s_{1,2} & \cdots \\ \vdots & \ddots & \\ s_{224,1} & & s_{224,d_j} \end{bmatrix}.$$

Next, we cut a random spectrogram segment S'_j from S_j such that its size is 224×224. In other words, we select a random $k \in \{1, 2, \ldots, d_j - 224\}$ and set

$$S'_j = \begin{bmatrix} s_{1,k} & s_{1,k+1} & \cdots \\ \vdots & \ddots & \\ s_{224,k} & & s_{224,k+224} \end{bmatrix}.$$

Pair (S'_j, y_j) consisting of the spectrogram segment and its label is the resulting training sample.

The process is repeated 8 times to populate the training batch. The batch is then fed into the neural network. The spectrograms corresponding to S'_j in the batch are used as the model input.

The CNN output can be interpreted as probability scores predicting that given audio recording x_j belongs to particular dance class. The labels corresponding to y_j are used for loss computation. Negative Log Likelihood Loss (NLL) [Zhu et al., 2018] is used as the loss function.

The model is trained to improve the prediction by repeating the training steps while new random batch is generated for each of the step. We rely on *Adam* optimizer [Kingma and Ba, 2014] with learning rate $lr = 0.0005$.

3.3 Segment Aggregation

While we train the network to predict short segments, it is beneficial to predict samples that are longer than the segment duration of 5.2 seconds. The reason is, that more information is naturally encoded in longer recordings, therefore classification accuracy can be higher. We accomplish this by classifying multiple segments belonging to given recording independently and aggregating their classification outputs.

Given an audio sample x and its spectrogram S, we cut the spectrogram to overlapping fixed size segments $S_1, S_2, \ldots, S_n$ with a given stride. Shorter stride leads to higher accuracy while longer stride leads to faster classification. We achieved good results with $stride = 200 \approx 4.6\,\text{seconds}$.

The CNN is executed for all segments $S_1, S_2, \ldots, S_n$. For a given segment S_i, the resulting softmax output $\boldsymbol{o}_i = model(S_i) = \{o_i^{(1)}, \ldots, o_i^{(10)}\}$ is a vector that represents probability scores that segment belongs to a particular dance class. Next, the resulting vectors $\boldsymbol{o}_1, \ldots, \boldsymbol{o}_n$ are averaged by arithmetic mean:

$$\boldsymbol{o} = \frac{\sum_{i=1}^{n} \boldsymbol{o}_i}{n} = \left(\frac{\sum_{i=1}^{n} o_i^{(1)}}{n}, \frac{\sum_{i=1}^{n} o_i^{(2)}}{n}, \ldots, \frac{\sum_{i=1}^{n} o_i^{(10)}}{n} \right).$$

The resulting vector $\boldsymbol{o} = \{o^{(1)}, \ldots, o^{(10)}\}$ represents probability scores that the audio sample x belongs to a particular class.

3.4 Discussion

This section describes arguments and justifications of design choices we made in the proposed method.

The proposed method is converting a raw audio recording to spectrogram image representation. Then, the spectrogram is cut to segments that are classified independently. This is in fact a sliding window classifier. The independent segment predictions are then aggregated.

We chosed the approach of classifying the segments independently, for a stationary nature of dance music. Dance music is a specific type of music containing repetitive patterns (beats, melody, ...) and stationary features (musical instruments, tempo, key, ...) that can be found and detected in each of the independent segments.

Popular means for sequence processing are recurrent neural networks (RNNs). However contrary to RNN that are suitable for time dependent problems as speech recognition, machine translation or action recognition. We suppose that each 5-seconds segment contains sufficient information for classification. This gives us a possibility to use CNN that is much faster than RNN and it is easier to train.

We can further benefit from averaging nature of aggregation, to classify dance music in real-time. The real-time classification can be achieved by aggregating the segments in incremental-average fashion. Precisely, we can classify newly-recorded segments independently, and update the aggregated average with each recent segment prediction, to achieve more accurate results.

The real-time classification of independent segments is shown in Figure 3.5. The figure represents probability scores $\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_n$ for consecutive segments

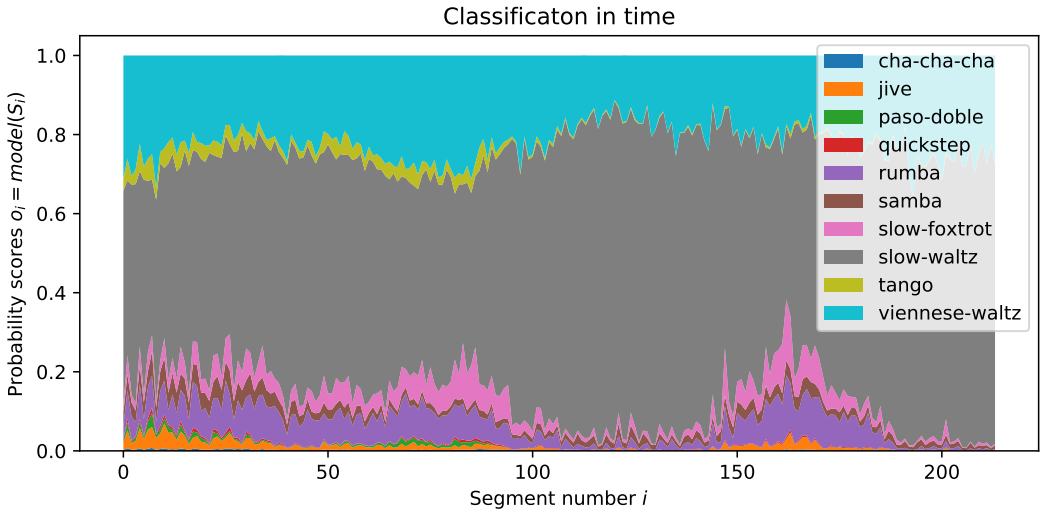

Figure 3.5: Classification of Slow Waltz. The independent classification of the segments in sliding window fashion allows us to predict the dance class in real-time. The figure shows probability scores for consecutive segments with $stride = 10$.

$S_1, S_2, \ldots, S_n$. Note, that the incremental average $\sum_{i=0}^{t} \boldsymbol{o}_i/t$ is visually represented by the areas corresponding to the dance classes.

Figure: stackplot - example

As the experiments will show in Section 4.4, the aggregation improves the classification results. We are aware of the fact, that output scores do not reflect the posterior probability distribution. It is well known that the neural networks often tend to be overconfident, i.e. the output is a very peaky distribution of scores. In our case, this effect is not that strong, as will be shown in Figure 4.13, that shows almost uniform distribution on the song beginning, where the classification is ambiguous. Nevertheless, we could still improve the aggregation by training a proper network confidence as proposed by Franc and Prusa [2019], which would probably further improve the aggregation results. Moreover, this way the reject option could be implemented, that allows the model to abstain from prediction in uncertain cases.

3.5 Implementation Details

3.5.1 Obtaining Audio Files

We trained and tested the model on several datasets. Python library *youtube_dl*[1] was used to download various samples from YouTube. The library was used in conjunction with *ffmpeg*[2] to convert the data to *wav* audio format. Each audio file contains name of corresponding YouTube video in its filename. We include scripts for easy downloading these datasets as an attachment.

[1] `https://github.com/ytdl-org/youtube-dl`

[2] `https://github.com/FFmpeg/FFmpeg`

3.5.2 Labeling Data

Next, our goal was to label the obtained audio files by splitting them to corresponding 10 dance classes.

We have chosen the class of each audio file by applying regular expression to given filename. As we have downloaded data from different YouTube channels, the regular expressions were different for each channel in order to be able to label as much files as possible. When a given filename matched two regular expressions corresponding to different dance classes, the file was marked as conflicting and was not added to the labeled dataset. Similarly, files with filenames matching no regular expression were not added to the dataset.

3.5.3 Spectrogram Computation

We have generated MEL spectrogram from audio signal of sampling rate 22050. *Librosa*[3], a Python library, was used to generate spectrogram with following parameters:

- Sampling rate $sr = 22050$
- Number of frequencies in y axis $n\_mels = 224$
- Time advance between frames $hop\_length = 512$
- Window size for STFT $n\_fft = 2048$
- Maximum frequency $fmax = sr/2 = 11025$

For audio recording with sampling rate $sr' \neq 22050$, we modify the parameters accordingly:

$$\begin{aligned} sr &= sr', \\ n\_fft &= \frac{2048 \cdot sr'}{22050}, \\ hop\_length &= \frac{512 \cdot sr'}{22050}. \end{aligned}$$

3.5.4 Removing Duplicate Samples

In order to train the model, we used our private dataset of 4655 labeled audio samples with ballroom dances. Then we tested the model using dataset obtained from YouTube as described above. This approach resulted into a problem with songs included in both training and testing dataset. Even though the corresponding audio signals were not equal and differed by distinct time shifts and audio quality, they both contained the same song recorded by the same interpreter.

Based on this insight, we introduce a script that identifies the near-duplicate audio files existing in both datasets. We used the script to delete the files from the train dataset. Next, we describe the technique used for identification of the near-duplicate files.

[3] `https://librosa.github.io`

Suppose we have two conflicting datasets D_1 and D_2 (in our case train dataset and test dataset) with labeled audio recordings (x_i, y_i). First, we convert each audio recording $x_i \in D_1 \cup D_2$ to a spectrogram

$$S_i = S(x_i) = \begin{bmatrix} s_{1,1} & s_{1,2} & \dots \\ \vdots & \ddots & \\ s_{224,1} & & s_{224,d_i} \end{bmatrix}$$

where d_i refers to spectrogram length.

Our goal is to find two audio samples x_i and x_j with spectrograms S_i and S_j resp. that correspond to the same song and $\exists y_i, y_j : (x_i, y_i) \in D_1 \wedge (x_j, y_j) \in D_2$. We accomplish this by looping over all spectrogram pairs S_i, S_j where each spectrogram belong to different dataset D_1 resp. D_2 and computing their correlation.

We define correlation of same-sized spectrograms represented by matrices $A, B \in \mathbb{R}^{n \times m}$ as

$$corr(A, B) = corr(a, b) = \frac{a^T \cdot b}{\sqrt{var(a) \cdot var(b)}}$$

where $a, b \in \mathbb{R}^{mn}$ are vectorized matrices elements. For different-width matrices $A, B : A \in \mathbb{R}^{n \times m}, B \in \mathbb{R}^{n \times m'}, m \neq m'$, we cut the overflowing matrix columns and proceed as in the same-size case.

Some of the recordings downloaded from YouTube has fixed music intro and the dance music starts after this intro. Based on this insight, we gradually shift spectrogram S_i along time axis by removing first k columns resulting into $S_i^{(k)}$. The correlation is computed for each of these shifts and the highest is chosen:

$$corr'(S_i, S_j) = \max_{k \in \{0, \dots, l\}} corr(S_i^{(k)}, S_j).$$

While we put $l = 200$ in our research, the spectrogram shifting allows us to match the audio signal with the correlated audio signal shifted in time.

The correlation higher than given threshold indicates, that the spectrograms correspond to the same recording. For our purposes, we have chosen $threshold = 0.7$, however the proper threshold may vary for different datasets.

Moreover, for the reason that comparing each spectrogram pair takes significant amount of computational power, we have compared only the pairs withing same dance class. This made the approach 10 times faster.

The technique of using spectrogram correlation has a potential flaw, that it does not detect recordings of the same song played by different interpreters since pitch and tempo can be different. So in theory, there might still be a small number of the same songs both in the training and the testing set. While we are aware of this flaw, we consider the problem of finding two distinct recordings of the same song challenging, hence we allow our model to take advantage of this flaw for predicting dance genre as part of its solution.

4. Experiments

4.1 Error Statistics

This section describes the error statistics that will be used to evaluate the experiments.

4.1.1 Accuracy

Accuracy, or **top-1 accuracy**, corresponds to percentage of test samples that were correctly classified. Accuracy of our model will be computed two times, once without the segment aggregation (Section 3.3), once with the aggregation included. In the first case, when **the segment aggregation is used**, an average accuracy over 5.2-seconds segments will be computed. In other case, when the **aggregation is used**, the accuracy from aggregated results will be computed for test samples corresponding to entire audio recordings.

4.1.2 Top-2 accuracy

Top-2 accuracy refers to percentage of cases, where either the first or the second predicted class with the highest probability score corresponds to the true class.

4.1.3 Confusion Matrix

A confusion matrix is represented by a matrix that describes relationships between true classes and predictions. Element a_{ij} of the confusion matrix represents a ratio of instances in class i, that were predicted as class j, versus the total number of instances in class i. Diagonal elements a_{ii} correspond to the accuracy of predicting class i.

4.2 Datasets

To train and test our model, we use datasets with ballroom dance music labeled by dance genres. In this section, we describe our private dataset used for training. Then, we describe datasets used for testing and validation, that are publicly available.

4.2.1 Training Dataset

We train our model with a private dataset that consists of 4655 audio recordings of ballroom dance music that belong to 10 dance classes. We collected this dataset from dance music albums of various interpreters. The audio is recorded in studio quality and each recording is about 4 minutes long.

Table 4.1 specifies number of recordings for each of the dance classes. Note that number of recordings corresponding to Paso Doble is small compared to other classes.

Dance Genre	**Count**
Cha Cha Cha	711
Jive	490
Paso Doble	112
Quickstep	458
Rumba	658
Samba	721
Slow Foxtrot	421
Slow Waltz	411
Tango	395
Viennese Waltz	281
Total	**4655**

Table 4.1: Number of recordings for each of the classes in the training dataset.

4.2.2 Testing and Validation Datasets

For testing and validation, we use dataset obtained from YouTube to minimize overlap with the training dataset. We created the dataset by extracting an audio track from videos in YouTube channel with ballroom music[1]. Moreover, the YouTube dataset is publicly available and can be used by other researchers to compare their results with our method. The YouTube dataset can be downloaded at `http://dance.ironbrain.net/testset.zip`. It is distributed as an archive with two text files corresponding to test and validation dataset. Each text file contains links to YouTube videos with ballroom music along with label for each link.

To ballance classes among the downloaded dance music, we selected 12 recordings for each of the dance class and split it uniformly to testing and validation datasets. Thus, both testing and validation datasets consist of 10 classes of 6

[1] `https://www.youtube.com/channel/UC0bYSnzAFMwPiEjmVsrvmRg`

recordings each which implies $10 \cdot 6 \cdot 2 = 120$ recordings in both testing and validation datasets.

The recordings are about 3 minutes long and they are in studio quality. Some of the recordings has fixed few-seconds intro, that was added to the dance music by authors of the channel. Since the intro does not relate to any dance genre and it is short compared to the rest of the recording, we suppose it has minor impact for classification and we keep it in the recording.

As described in Section 3.5.4, we verified that recordings in the test and the validation datasets do not overlap with recordings in the training dataset. Further, in order to replicate the test and the validation dataset, we include a script for extracting the audio from the YouTube channel and for labelling them by dance classes with use of corresponding video titles.

4.2.3 Extended Ballroom Dataset

To further validate our results and compare it with other approach as Marchand and Peeters [2016b], we use publicly available dataset Extended Ballroom [Marchand and Peeters, 2016a], as described in Section 2.3.1. The Extended Ballroom dataset contains ten-seconds audio recordings in 13 dance classes as shown in Table 4.2.

Dance Genre	**Count**
Cha Cha Cha	455
Jive	350
Quickstep	497
Rumba	470
Samba	468
Tango	464
Viennese Waltz	252
Waltz	529
Foxtrot	507
Paso Doble	53
Salsa	47
Slow Waltz	65
West Coast Swing	23
Total	**4180**

Table 4.2: Number of recordings for each of the classes in the Extended Ballroom dataset [Marchand and Peeters, 2016a].

To use the Extended Ballroom dataset for testing of our method with ten classes, several modification were performed. Firstly, classes Salsa and West Coast Swing were removed from the dataset. Since the audio recordings in Extended Ballroom's classes Waltz and Slow Waltz correspond to one class in our dataset, the both classes were merged resulting to 594 audio recordings of Slow Waltz. Finally, class Foxtrot was renamed to Slow Foxtrot. The resulting dataset consists of 4110 audio recordings as shown in Table 4.3.

Dance Genre	**Count**
Cha Cha Cha	455
Jive	350
Paso Doble	53
Quickstep	497
Rumba	470
Samba	468
Slow Foxtrot	507
Slow Waltz	594
Tango	464
Viennese Waltz	252
Total	**4110**

Table 4.3: Number of recordings for each of the classes in the modified Extended Ballroom dataset.

4.3 Baseline Algorithm

To assess the performance improvement that can be achieved by the CNNs, we also train a simple machine learning classifier that rely on hand-crafted audio features [Bahuleyan, 2018]. In this section, we describe such baseline classifier and its results.

4.3.1 Hand-Engineered Features

For each audio recording, we extract hand-crafted features bellow. We have chosen representative features from both time domain and frequency domain that are used by Bahuleyan [2018]. A vector, that is created by concatenating all the features is referred as a feature vector. The features were extracted using a Python library *Librosa*[2].

- **Zero Crossing Rate (ZCR)** - This feature corresponds to a number of points where the signal changes sign from positive to negative [Gouyon et al., 2000]. We divided the whole recording to small overlapping windows 2048 samples long with a hop size 512 (note that we used these parameters across other features as well). Then, ZCR was computed for each of the window and the average along with the standard deviation of all ZCRs were added to the feature vector.

- **Chromagram** - This feature refers to the energy of the signal in each of the pitch classes as C, C#, D, D#, E, F, F#, G, G#, A, A# and B [Ellis, 2007]. This vector is computed for all of the windows and average and standard deviation for each of the pitch are chosen as representative features.

- **Spectral Centroid** Spectrum is computed for each window and treated as a distribution over frequency bins. The distribution mean is computed and referred as centroid [Klapuri and Davy, 2007].

  More precisely, the centroid for window t is defined as

  $$f_t^{(c)} = \sum_k \frac{S(k,t)f(k)}{\sum_j S(j,t)}$$

  where $S(k,t)$ is a spectral magnitude of frequency bin k in window t, and $f(k)$ is a frequency corresponding to the bin k. Note that function S can be represented by a spectrogram.

  Centroid $f_t^{(c)}$ is computed for each of the windows and mean and standard deviation are added to the feature vector.

- **Spectral Band-width** While spectral centroid corresponds to distribution mean, 2nd order spectral band-width represents standard deviation [Klapuri and Davy, 2007].

[2] `https://librosa.github.io`

- **Spectral Roll-off** The roll-off frequency [Klapuri and Davy, 2007] is computed for each window and corresponds to the frequency f such that frequencies $f' : f' \leq f$ contain 85% (this threshold can be changed) of the total energy in the spectrum. It can be used to approximate the maximum frequency by setting the threshold to a value close to 1.

- **Mel-Frequency Cepstral Coefficients (MFCC)** This feature has been very useful for tasks as speech recognition [Davis and Mermelstein, 1980] and it is represented by a vector of size 20 (this value can be changed) that is computed for each window. Again, the mean and standard deviation is computed among the windows, which results to a vector of size 40 that is added to the final feature vector.

The dimension of the resulting feature vector of the concatenated features is 72.

4.3.2 Classifier

The feature vectors extracted from audio files are used to train support-vector machine (SVM) classifier ([Cortes and Vapnik, 1995]. A radial basis function (RBF) kernel is used because it is required to address this non-linear problem.

4.3.3 Baseline Algorithm Results

The SVM classifier was trained using the training dataset (Section 4.2.1) resulting in accuracy **40%** on test data (Section 4.2.2). Figure 4.1 shows the confusion matrix of the resulting classifier.

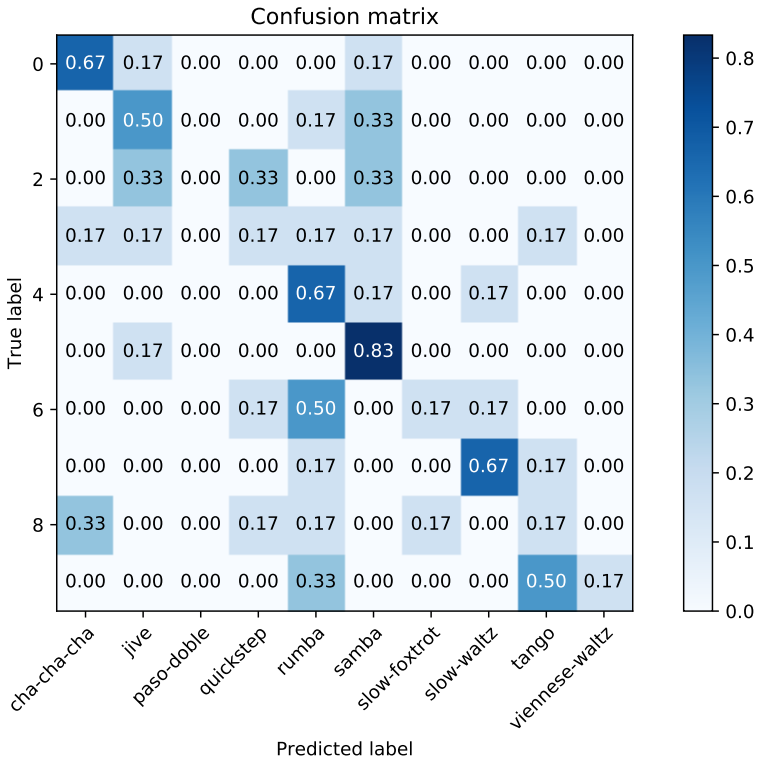

Figure 4.1: Confusion matrix of the baseline algorithm, a SVM classifier that took handcrafted features, each created from one audio recording, as an input. The confusion matrix was calculated from results on testing dataset.

4.4 Results

In this section, our model will be evaluated. The model was trained using the training dataset (Section 4.2.1) and it will be evaluated using the testing and validation datasets (Section 4.2.2), that are publicly available.

The model is trained by epochs of 20 batches each consisting of 8 spectrogram segments. For each epoch, training loss and accuracy are calculated. Losses and accuracies for the test and the validation dataset are calculated every twentieth epoch for the long calculation process. The model was trained using 2000 training epochs, as seen in Figure 4.2. The segment aggregation is not employed in this case.

The last training epochs (1500 - 2000) show, that the testing and validation loss is higher than the training loss and similarly the testing and validation accuracy is lower than the training accuracy. This is caused by model overfitting which is reduced by dropout [Srivastava et al., 2014] as described in Section 3.2.1. The dropout is properly used only on training data which is seen in the majority of first epochs (0 - 500), where the model performs better on test and validation data.

To further improve our results, the segment aggregation is employed and the accuracy is calculated from the aggregated results. We calculate the accuracy after the aggregation for both train and validation data every twentieth epochs. As shown in Figure 4.3, the accuracy increases when the aggregation is employed.

The model corresponding to the epoch with the highest accuracy on the validation data is finally selected for evaluation. The epoch with the highest accuracy on the validation data will be referred as the **best epoch**. The accuracy on the validation data is computed without the segment aggregation employed for finer scale of the results because aggregation leads to accuracy 100% for some epochs. The model corresponding to the best epoch is then evaluated using the test dataset.

On the test dataset, the model achieved results shown in Table 4.4.

Method	**Top-1 accuracy**	**Top-2 accuracy**
Our method with aggregation	96.7%	100.0%
Our method without aggregation	92.2%	-

Table 4.4: Our results on the test dataset.

Note, that the model training is not deterministic and each training can lead to a different accuracy. The variance of such accuracies can be high because the test dataset of 60 audio recordings is relatively small. This motivates us to further test our model on the modified Extended Ballroom dataset (Section 4.2.3) which is larger and allows us to compare our method with other method, MASSS [Marchand and Peeters, 2016b]. The authors of MASSS, that stands for Modulation Scale Spectrum with Auditory Statistics, mainly focus on representation of audio content. They present their technique on a simple classifier, that our results are be compared with, but such classifier is marginal and not fully explained.

The results of our model on the Extended Ballroom dataset, consisting of 4,000+ audio recordings each 10 seconds long, are shown in table 4.5.

While the MASSS accuracy is slightly higher than the accuracy of our method, one should consider that the number of classes used by the MASSS approach is lower (9 instead of 10). Moreover, the testing configuration of the MASSS is uncertain, because the authors did not specify what data they use for the training and the testing protocol is unclear.

Moreover, the results achieved on the test dataset are better than the results achieved on the Extended Ballroom dataset not only, because more information is naturally encoded in longer audio recordings of the test dataset, but also because each ten seconds recording of the Extended Ballroom dataset represents beginning of a dance song. Such recording often corresponds to a song intro whose prediction is ambiguous.

Method	**Top-1 accuracy**	**Top-2 accuracy**
MASSS [Marchand and Peeters, 2016b]	94.9%[a]	-
Our method with aggregation	93.9%	97.5%
Our method without aggregation	86.6%	-

Note: [a] While this accuracy is slightly higher than the accuracy of our results, one should consider that the number of classes used by the MASSS approach is lower (9 instead of 10) and the testing protocol is unclear.

Table 4.5: Our results on the Extended Ballroom dataset compared to the MASSS method [Marchand and Peeters, 2016b], which is presented in italic.

4.4.1 Confusion Matrix

Next, confusion matrices of the model will be described. Figure 4.4 shows the confusion matrices of our model evaluated on the test dataset. While prediction of most dances is accurate, the confusion matrices show Slow Waltz to be falsely classified as Viennese Waltz. These dances are challenging to predict, since they are related to each other and with highly similar patterns. Note, that the values in the confusion matrix of the method with the segment aggregation are rather round compared to the values in the confusion matrix of the method without the aggregation, because by design, the segment aggregation reduces number of results.

The confusion matrices of our model evaluated on the Extended Ballroom dataset are shown in Figure 4.5. We highlight dance Rumba that is mostly misclassified as Slow Waltz or Slow Foxtrot for the slow nature of these three dances.

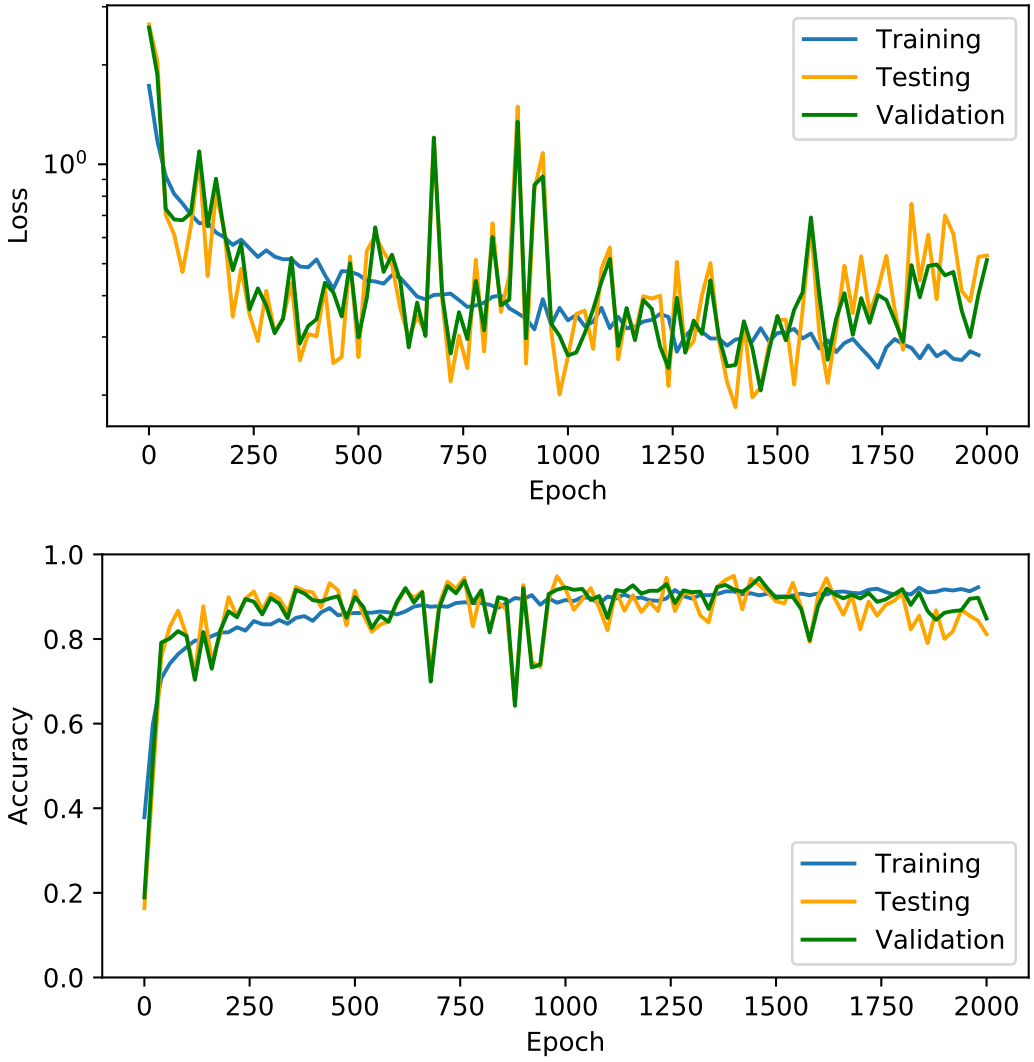

Figure 4.2: Training of our model. Values of the loss (top) and the accuracy (bottom) when training our model. The accuracy is computed for the outputs without the segment aggregation employed.

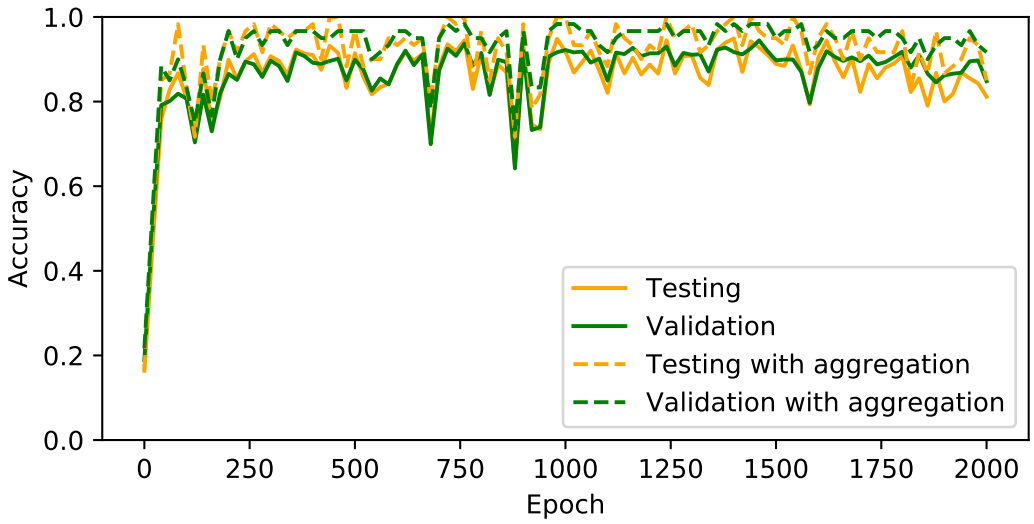

Figure 4.3: Comparison of the model accuracy when the segment aggregation is not employed (solid lines) and with the segment aggregation employed (dashed lines). The accuracy is computed using the test and the validation datasets.

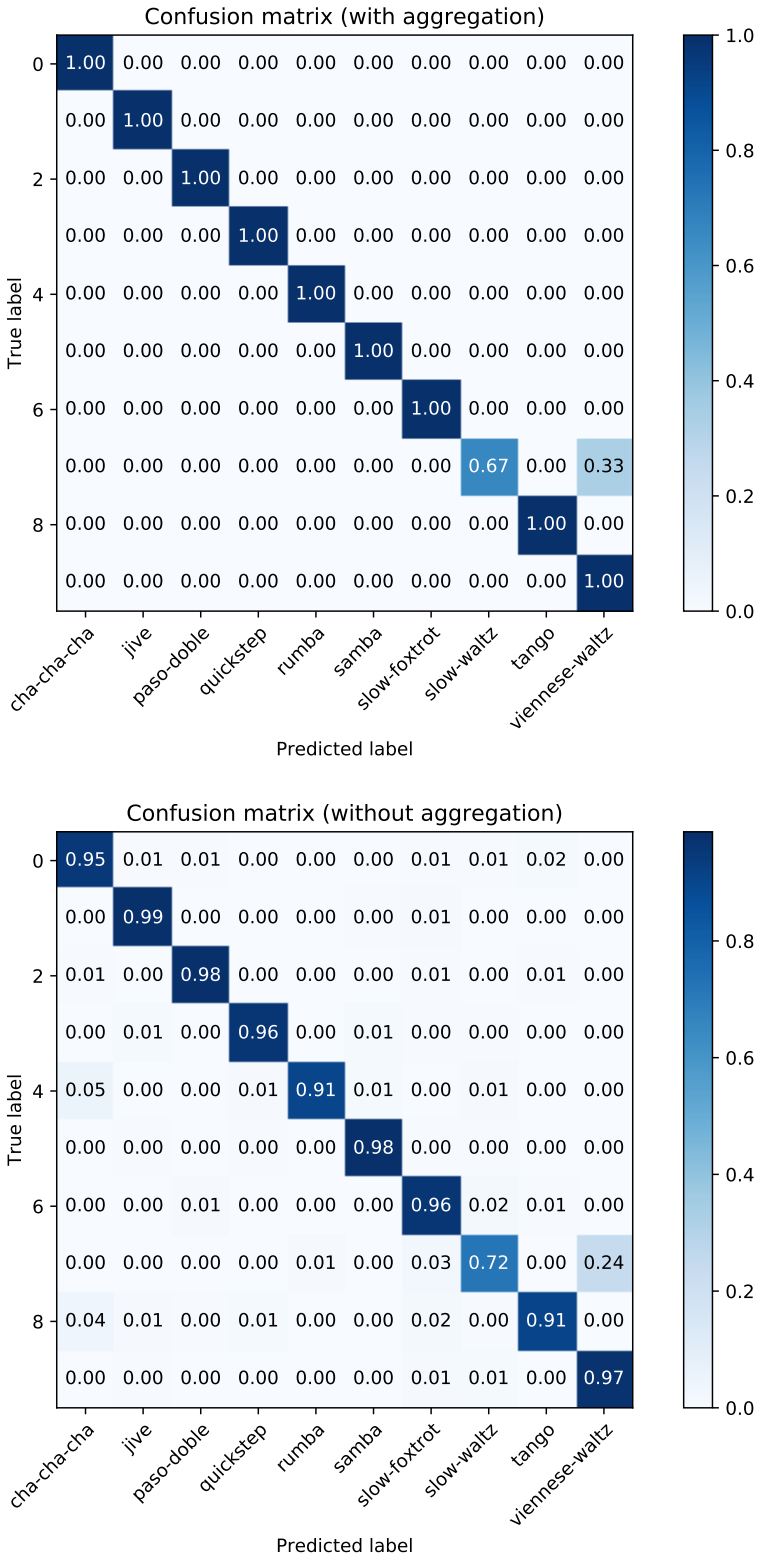

Figure 4.4: Confusion matrices of our model on the test dataset for both with (top) and without (bottom) the segment aggregation.

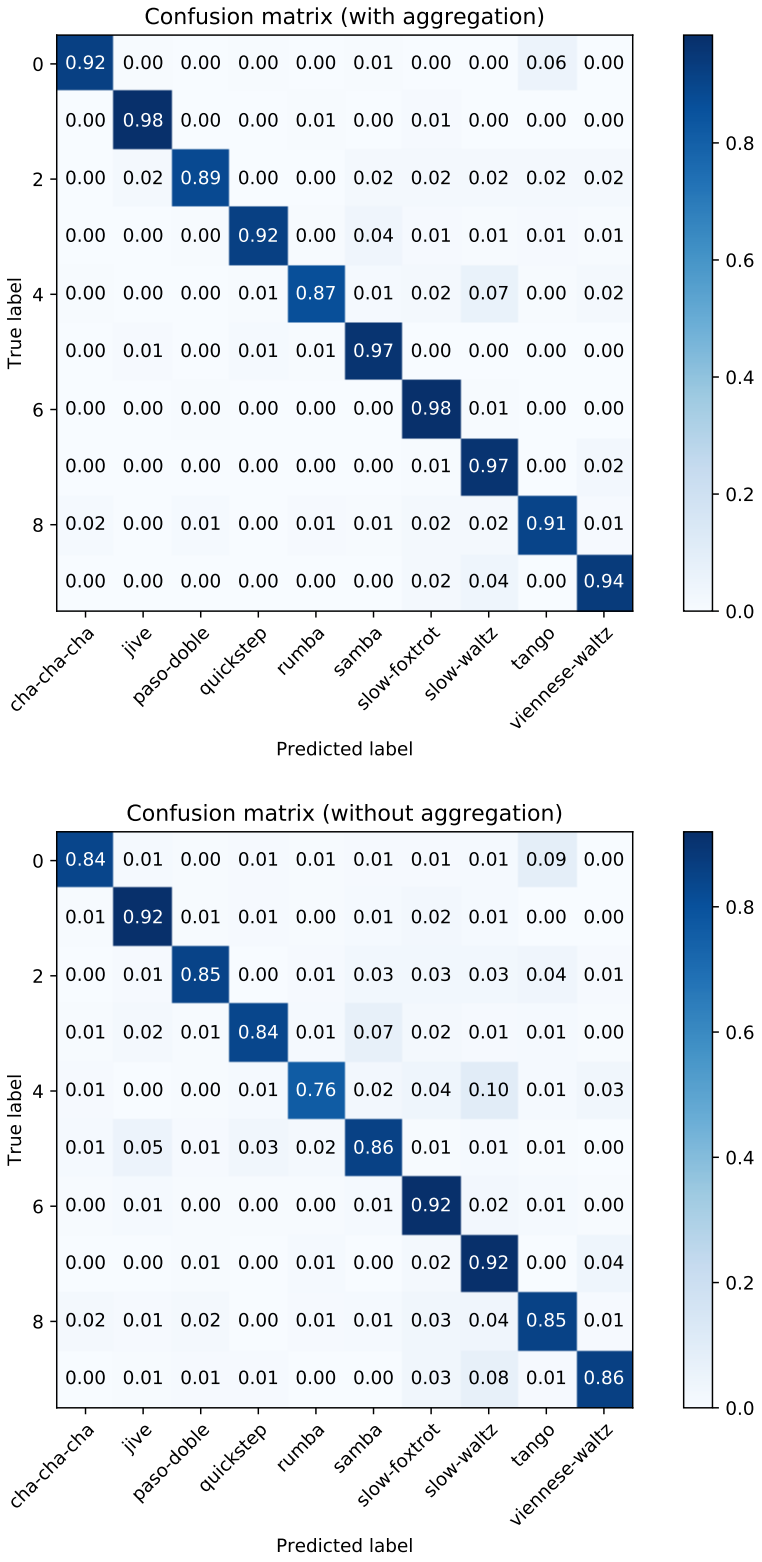

Figure 4.5: Confusion matrices of our model on the Extended Ballroom dataset for both with (top) and without (bottom) the segment aggregation.

4.5 CNN Models Comparison

In this section, we compare DenseNet [Huang et al., 2017] with other CNN architectures as ResNet [He et al., 2016]. Moreover, to empirically demonstrate effectiveness of our method that uses pre-trained model and trains all its parameters, we compare this approach with other approaches as using transfer learning techniques, starting with random parameters and modifying DenseNet architecture.

4.5.1 Comparison of Different CNN Architectures

As described in Section 3.2.1, CNN network DenseNet 161 [Huang et al., 2017] is used in this study. We compare this CNN architecture with other three CNN architectures as VGG 16 [Simonyan and Zisserman, 2014], ResNet-18 [He et al., 2016] and ResNeXt-50 32x4d [Xie et al., 2017]. The described architectures are trained using the training set (Section 4.2.1) and similarly to our method, pre-trained networks are used with parameters learned on ImageNet [Deng et al., 2009]. The last classification layer is replaced with layer of 10 neuron outputs which correspond to the number of dance classes.

As seen in Table 4.6, DenseNet outperforms other CNN architectures. While there is a significant difference between the accuracy of VGG and DenseNet, the accuracies of novel architectures DenseNet, ResNet and ResNeXt are similar.

Architecture	**Top-1 accuracy**	**Top-2 accuracy**	**Top-1 without aggregation**
VGG 16	25.0%	41.7%	24.8%
ResNet-18	**96.7%**	**100.0%**	89.9%
ResNeXt-50 32x4d	95.0%	**100.0%**	89.6%
DenseNet 161	**96.7%**	**100.0%**	**92.8%**

Table 4.6: Accuracies on the test dataset for given CNN architectures. The highest values are **bolded**.

4.5.2 Comparison of Training Configurations

Since DenseNet 161 is the best performing CNN architecture for our task, we will use it to further experiment with its training process configuration. The configurations we will experiment with can be broadly separated to two groups. The first group (DenseNet-TL) uses pre-trained DenseNet with parameters learned on ImageNet. It enables us to use transfer learning (TL) approaches. Other group (DenseNet-RW) uses DenseNet's parameters initialized with random values which allows us to further modify the DenseNet's structure without loosing information about pre-trained parameters as it would happen with pre-trained approach.

Since the number of DenseNet's classification classes is 1000 by default, we replace the classification linear layer of DenseNet with linear layer of 10 output neurons which correspond to number of dance classes. The parameters of such layer cannot be learned on ImageNet dataset due to the number of the dataset classes and they are initialized with random values.

In our experiments, the DenseNet will be trained with the following configurations.

DenseNet-TL-C In this transfer learning configuration, all convolution layers are frozen and only the classification layer, which is represented by densely connected layer of size 2208×10, is trained.

DenseNet-TL-DB4 (half or full) Not only the classification layer, but also the last of the four dense blocks is trained in this configuration. The 4th dense block is initialized with parameters learned on ImageNet as well as other dense blocks. In our experiments, either second half of the dense block (12 convolution layers) is trained, or the full dense block (24 convolution layers) is trained, respectively.

DenseNet-TL-DB4-N To increase the capacity of the trained neural network parameters, we replace the last dense block with a dense block of higher size. Similarly to DenseNet-TL-DB4, the parameters of the first three dense blocks are frozen. The last dense block is replaced with a dense block of n convolution layers for given n. The newly created dense block has the same *growth rate* as the substituted dense block and its parameters are initialized with random values. Note, that for increasing n, the number of input neurons in classification layer increases by design.

DenseNet-FT This is our proposed configuration (Section 3.2.1) where pre-trained DenseNet with parameters learned on ImageNet is employed.

DenseNet-RW This configuration employs DenseNet with all parameters initialized with random values in order to demonstrate effectiveness of the approach in configurations above, where pre-trained DenseNet with parameters learned on ImageNet was employed. In this configuration, DenseNet 161 is used with classification layer modified only and the model is initialized with random parameters.

DenseNet-RW-1C7x7 While DenseNet is designed for classification of RGB images, its first convolution layer has three input channels. To improve the model performance on one-channel spectrograms, we replaced the first three channel convolutions with convolutions of one input channel, while keeping the kernel size 7×7, and feed the spectrogram segments to the network accordingly.

DenseNet-RW-1C16x3, DenseNet-RW-1C40x3 To highlight the importance of vertical axis of spectrograms, where the vertical axis represents the spectrum of frequencies, we replace 7×7 convolutions with vertical convolutions of sizes 16×3 and 40×3 resp. Similarly to DenseNet-RW-1C, the convolutions have one input channel.

Results

Table 4.7 shows that training of all network parameters (*DenseNet-FT, DenseNet-RW*) significantly outperforms the approaches where some parameters are frozen

(*DenseNet-TL*). The low results of configurations with frozen parameters are not surprising. The statistics of spectrograms (and consequently the learned convolutional filters) are very different than the image statistics on the ImageNet dataset. However, to freeze some model parameters significantly reduce the number of trainable parameters and can be beneficial for training on a small dataset.

While both, approach of using pre-trained model and the approach of using the randomly initialized model, achieve similar results, modification of the first convolution layer to have one input channel does not significantly improve the results. Although, it is not very surprising, because the capacity of the modified convolution layer consisting of $3 \cdot 7 \cdot 7 \cdot 96 = 14112$ parameters is insignificant compared to the capacity of whole model. Moreover, the vertical convolutions decreased the accuracy rather than helping with the classification.

Configuration	**Top-1 accuracy**	**Top-2 accuracy**	**Top-1 without aggregation**
DenseNet-TL-C	63.3%	76.7%	42.9%
DenseNet-TL-DB4 (half)	80.0%	85.0%	62.7%
DenseNet-TL-DB4 (full)	83.3%	91.7%	64.5%
DenseNet-TL-DB4-N (n=24)	70.0%	85.0%	62.0%
DenseNet-TL-DB4-N (n=48)	76.7%	80.0%	63.3%
DenseNet-TL-DB4-N (n=72)	75.0%	86.7%	64.5%
DenseNet-FT	**96.7%**	**100.0%**	**92.8%**
DenseNet-RW	95.0%	**100.0%**	91.3%
DenseNet-RW-1C7x7	93.3%	**100.0%**	89.2%
DenseNet-RW-1C16x3	93.3%	**100.0%**	88.2%
DenseNet-RW-1C40x3	91.7%	98.3%	88.2%

Table 4.7: Accuracies on the test dataset for given training configurations. The highest values are **bolded** as well as our proposed configuration.

4.6 Cross-dataset Testing

In this section, we will test our model (see Section 4.4) performance on datasets of various quality. The model will be evaluated using 4 different datasets. We will compare the method on a dataset created from the same source as the training set (Section 4.2.1) with other datasets created from different sources. First of the different-source datasets is created from audio recordings of ballroom dance competitions. Other dataset is created using recordings from StarDance, Czech version of Dancing with the Stars TV show, where often popular songs are played instead of typical dance music. Finally, low quality audio from various dance competitions recorded using mobile phone, will show, how the model behave on noisy input.

4.6.1 Dataset from the Same Source as the Training Data

While we use a dataset consisting of only 60 audio recordings for testing (Section 4.2.2), it is beneficial to further evaluate our results using other, larger dataset.

Hence, when the private training dataset (Section 4.2.1) was created, 10 audio recordings were separated from each of 10 classes while creating new testing dataset. The created dataset contains 100 audio recording of ballroom dances and does not overlap with the training dataset.

4.6.2 Dataset from Dance Competitions

We leverage the popularity of dance competitions to create another dataset, that will be described in this section. It was created by extracting music from 363 YouTube dance videos[3]. Videos from various dance competitions of the World DanceSport Federation[4] (WDSF) were used. Both latin and standard dances are included.

Videos from the following competitions were used:

- WDSF GrandSlam Standard in Moscow (Russia), 2019
- WDSF GrandSlam Standard in Rimini (Italy), 2019
- WDSF European Championship Standard in Salaspils (Latvia), 2019
- WDSF GrandSlam Latin in Moscow (Russia), 2019
- WDSF GrandSlam Latin in Rimini (Italy), 2019
- WDSF European Championship Latin in Paris (France), 2019

4.6.3 StarDance

For the increasing popularity of TV show called StarDance, we evaluate the model using audio recordings extracted from the show. StarDance is a Czech dance competition, where couples (a celebrity + professional dancer) dance ballroom

[3] `https://www.youtube.com/user/DanceSportTotal/`

[4] `https://www.worlddancesport.org/`

dances, similarly to the show Dancing with the Stars popular in UK. While both latin and standard dances are competed, the background music is not typical dance music, but rather a popular music, that was not composed for a specific dance. This makes the dance recognition from such audio tracks challenging even for humans.

We created the dataset by extracting an audio from 70 videos of dances from 10th season[5] broadcasted in 2019. The StarDance dataset does not contain any recordings of Slow Foxtrot.

4.6.4 Low Quality Recordings

In order to test the model robustness to noise and low quality input in general, we extracted audio from 85 YouTube videos of dance competitions, that were recorded using a mobile phone camera. The audio quality of such recordings is very low, including echo, people applauding, dancers steps sounds, and other noise and audio artifacts.

4.6.5 Results

As expected, the introduced test dataset, that was created by separating 100 audio recordings from the training dataset, has similar results as the original testing dataset, as seen in Table 4.8.

Dataset	**Top-1 accuracy**	**Top-2 accuracy**	**Top-1 without aggregation**
YouTube test dataset	96.7%	100.0%	92.2%
Separation from training dataset	93.0%	98.0%	86.5%
Dance competitions	87.9%	98.6%	70.6%
StarDance	68.0%	78.0%	45.2%
Low Quality Recordings	72.7%	86.7%	58.0%

Table 4.8: Results of our method on various datasets.

Compared to the testing dataset, the accuracy of the dataset with dance competitions is slightly lower. We suggest an explanation, that contrary to the testing dataset, the audio recordings are recorded using a microphone placed in the dancing hall and, apart from the dance music, audio contains noise as steps of the dancers in the background. Note, that the top-2 accuracy of the dance competitions dataset is similar to the top-2 accuracy of the test dataset.

Our model performed unsurprisingly worse on the StarDance dataset. The accuracy is low because the dataset contains popular songs rather than a typical ballroom music and the classification is ambiguous even for humans.

As shown in Figure 4.6, the songs played for Jive dance in StarDance TV show sound like Jive music. On the other hand, there are not many popular songs used that sound like Paso Doble. Further, since Slow Waltz and Viennese Waltz dances are accompanied by triple-meter music only and cannot be accompanied by

[5] `https://www.ceskatelevize.cz/porady/12607522764-stardance-x/`

music for other dances, both Slow Waltz and Viennese Waltz are not significantly confused with other dances.

The worst results is seen in low quality recordings. It is not surprising because the model was trained using studio-quality audio with no noise or artifacts and is sensitive to low quality data. This problem will be addressed in Section 4.7.

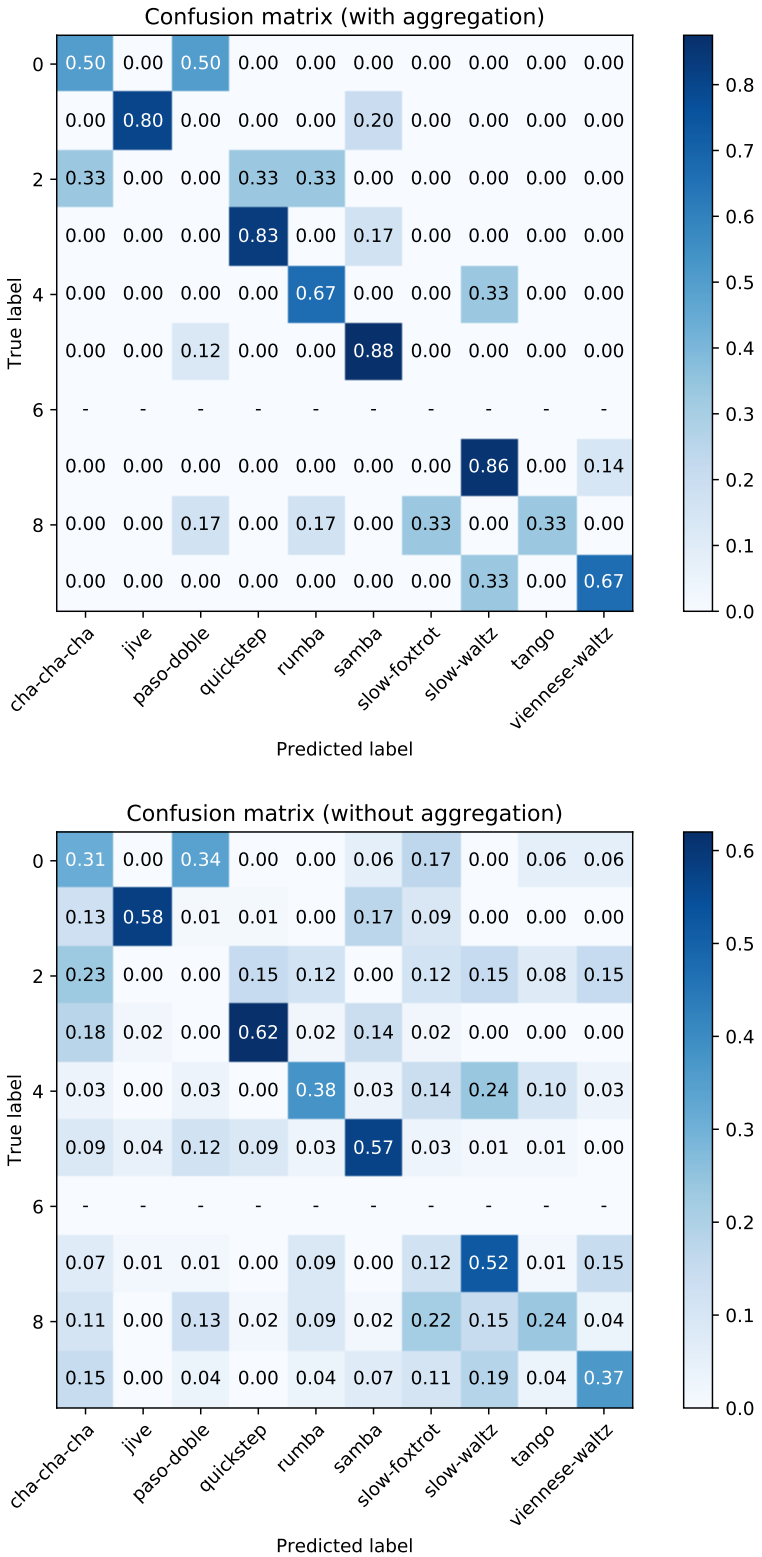

Figure 4.6: Confusion matrices of our model on the StarDance dataset for both with (top) and without (bottom) the segment aggregation. The classification on the StarDance dataset is ambiguous because the dataset contains popular songs rather than a typical ballroom music. Empty cells correspond to the Slow Foxtrot, which is not contained in the StarDance dataset.

4.7 Sensitivity to Low Quality Data

While our model performs well on audio recordings of studio quality, it has difficulties with low quality data. As Table 4.8 has shown, the model achieved accuracy of only 72.7% on the low quality recordings.

To further illustrate the model sensitivity on data with background noise, we mixed a recording of crowd noise, extracted from YouTube video[6], into each audio recording in test dataset with given intensity α.

The perturbed signal is the convex combination in the temporal domain:

$$\widetilde{x}(t) = (1 - \alpha)x_{signal}(t) + \alpha x_{noise}(t)$$

where x_{signal} corresponds to the original signal and x_{noise} corresponds to the signal of the noise recording. We call the resulting set of audio recordings created by mixing the noise into the recordings in the test set, as the synthetic test set.

Both signals, the original from the test set and the crowd noise, are normalized so that:

$$\max x(t) = 1$$

where $x(t)$ corresponds to the value of signal sample for given time t.

To illustrate the intensity of the normalized signals, root mean square (RMS) is calculated as:

$$RMS = \sqrt{\frac{1}{n}\sum_{t}\Big(x(t)\Big)^2}$$

where n is the number of the samples. The resulting RMS for normalized noise recording was calculated as $RMS_{noise} = 0.16$ and the averaged RMS of all normalized recordings from the test set was calculated as $RMS_{signal} = 0.23$.

We tested the model on the synthetic test set, each with increasing background noise intensity. The results are shown in Figure 4.7.

We describe the ratio of the original signal intensity and the intensity of the noise using signal-to-noise ratio (SNR). SNR of the synthetic test set with background noise of intensity $\alpha = 0.5$ is:

$$SNR = \frac{RMS^2_{signal}}{RMS^2_{noise}} = 1.95$$

Figure 4.8 shows an example of synthetically adding noise to the Jive, audio recording from the test set, with ratio $\alpha = 0.5$.

4.7.1 Extending the Training Dataset

To improve the model accuracy on low quality data, we added low quality audio recordings to the training dataset. These recordings are similar to those low quality recordings used for testing, as they are extracted from the same YouTube channel. Although the both datasets do not overlap. The number of low quality

[6] `https://www.youtube.com/watch?v=IKB3Qiglyro`

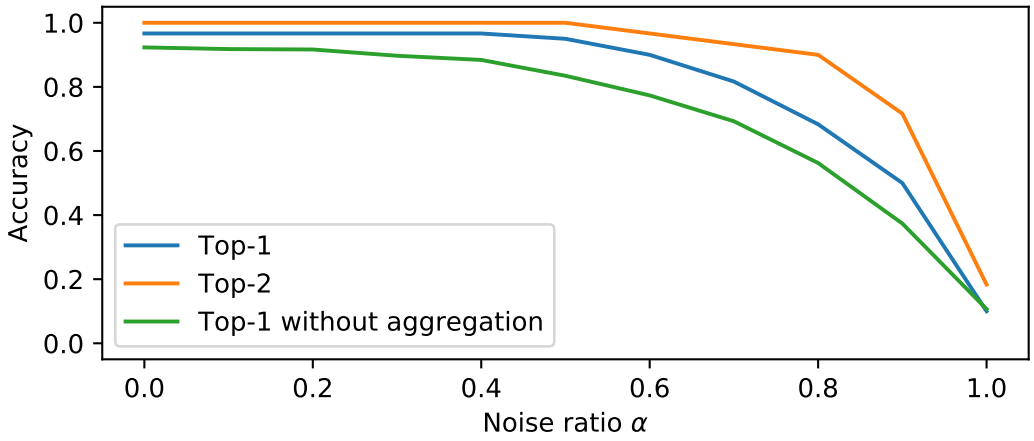

Figure 4.7: Accuracy on synthetic test set with background noise of increasing intensity, using our original model. The synthetic test set was created by adding crowd noise to the recordings from the testing dataset in given ratio α. Ratio 0.0 corresponds to the original audio recording, ratio 1.0 correspond to a recording of the noise only.

Dataset	**Top-1 accuracy**	**Top-2 accuracy**	**Top-1 without aggregation**
YouTube test dataset	**98.3%**	100.0%	**92.7%**
Separation from training dataset	93.0%	98.0%	86.0%
Dance competitions	**93.7%**	**98.9%**	**75.7%**
StarDance	46.0%	74.0%	38.6%
Low Quality Recordings	**89.8%**	**95.3%**	**71.7%**

Table 4.9: Results of a model, where low quality audio recordings were added to the training dataset. The **bold** values represent accuracies that increased compared to the previous results in Table 4.8.

recordings added to the training set is 239 which corresponds to 4.88% of audio recordings in training set overall.

Then, new model is trained on the extended dataset from the training dataset. Similarly to the original model, the pre-trained DenseNet with parameters learned on ImageNet [Deng et al., 2009], is used.

4.7.2 Results

The resulting model has achieved substantially better performance on low quality data, than the old model has (Table 4.9). The resulting accuracy increased by $89.8\% - 72.7\% = 17.1$ p.p. on low quality recordings dataset. Moreover, the accuracy on dance competitions dataset has increased. The reason is probably that the dance competitions have certain amount of background noise that the extended model can handle. Decreased accuracy is seen in StarDance dataset and it decreased by $68\% - 46\% = 12$ p.p. The accuracy decrease of the StarDance dataset is mostly caused by ambiguity of the StarDance dataset which is indicated by the top-2 accuracy with slight decrease of $78\% - 74\% = 4$ p.p.

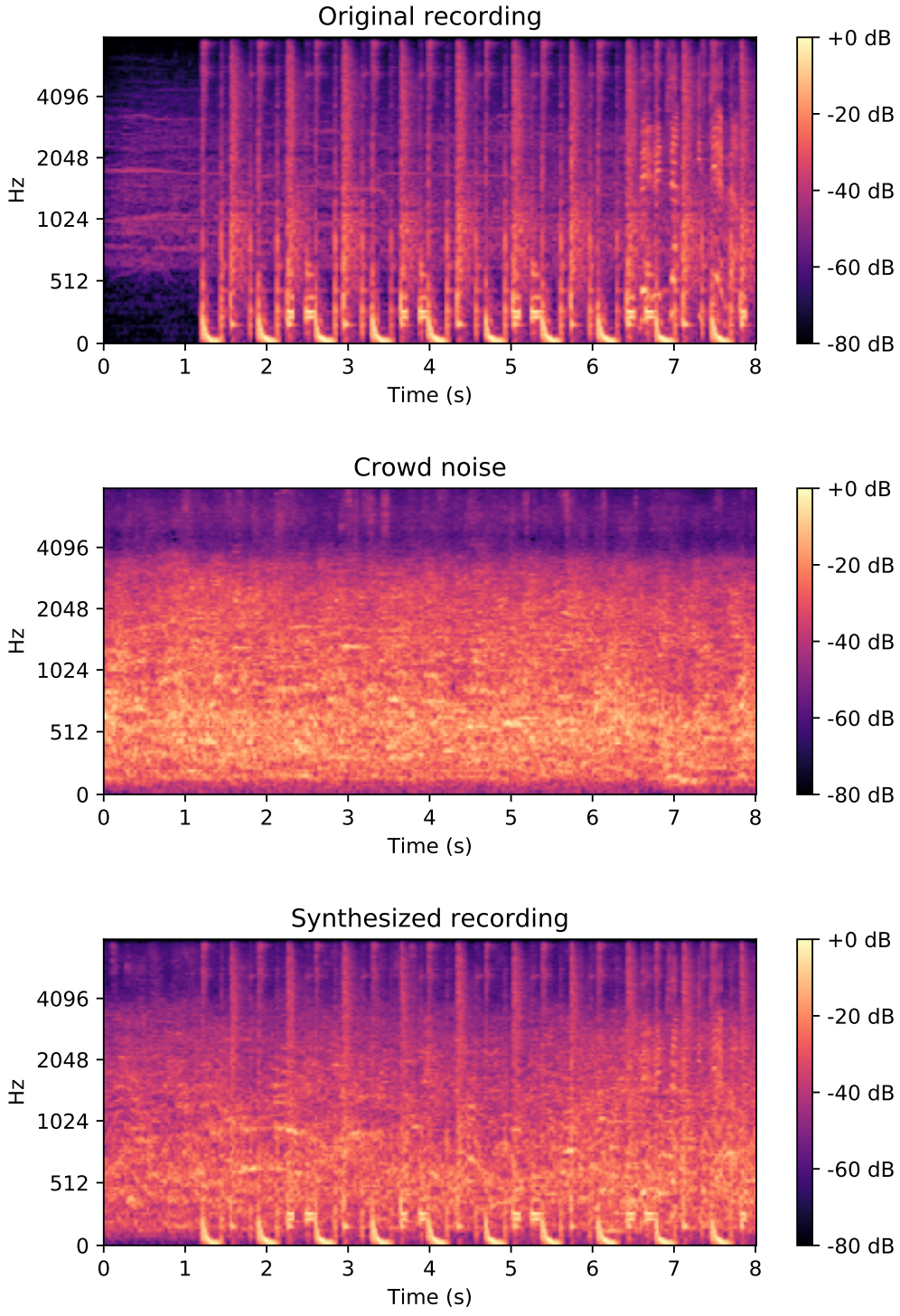

Figure 4.8: The recording of Jive (first) is mixed with the recording of crowd noise (second) resulting in the synthesized recording (third). The corresponding mel spectrograms are shown and the mixing ratio $\alpha = 0.5$.

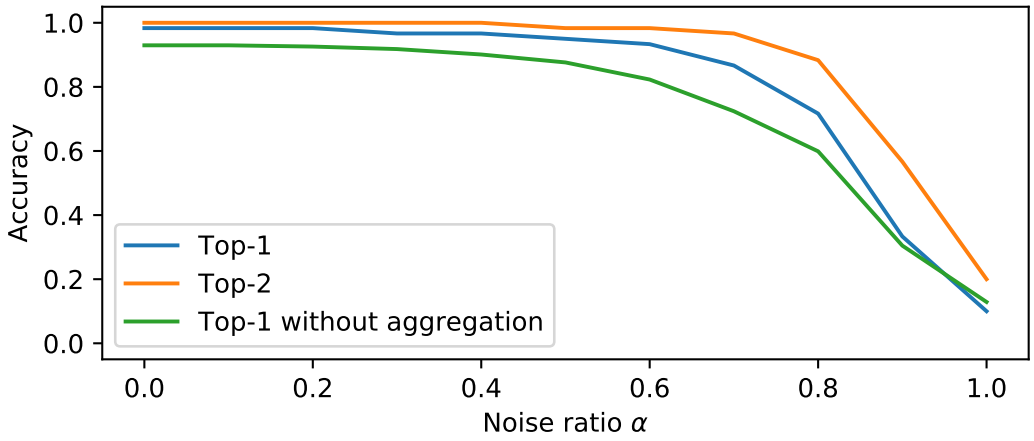

Figure 4.9: Accuracy on synthetic test set with background noise of given intensity, using a model, that was trained the extended training set. See the improvement compared to Figure 4.7. The synthetic test set was created by adding crowd noise to the recordings from the testing dataset in given ratio α. Ratio 0.0 corresponds to the original audio recording, ratio 1.0 correspond to a recording of the noise only.

To further investigate the suggested model results, we test the model on the synthetic test set with increasing crowd noise intensity. Figure 4.9 shows slight improvement compared to the results of previous model.

In conclusion, we have achieved substantially better results on low quality data with simple technique of adding noisy recordings to the training dataset in a ratio as small as 4.9%.

4.8 Classification Examples

This section shows remarkable classification outputs of chosen audio recordings from both the test set and the StarDance dataset. Our proposed model is used for the classification. All mentioned audio recordings are publicly available and links referring to them are included. Moreover, we propose a web application as demonstration of our approach.

4.8.1 Demonstration

Our method is demonstrated using a responsive web application at `http://dance.ironbrain.net` as shown in Figure 4.10. The application enables user to upload audio file and shows the classification results to the user. The classification results are represented by probability scores over the dance classes. The classification is performed using the model, that is trained on the training set extended by low quality audio recordings, as described in Section 4.7. The demonstration achieves promising results and it has a potential for a commercial deployment.

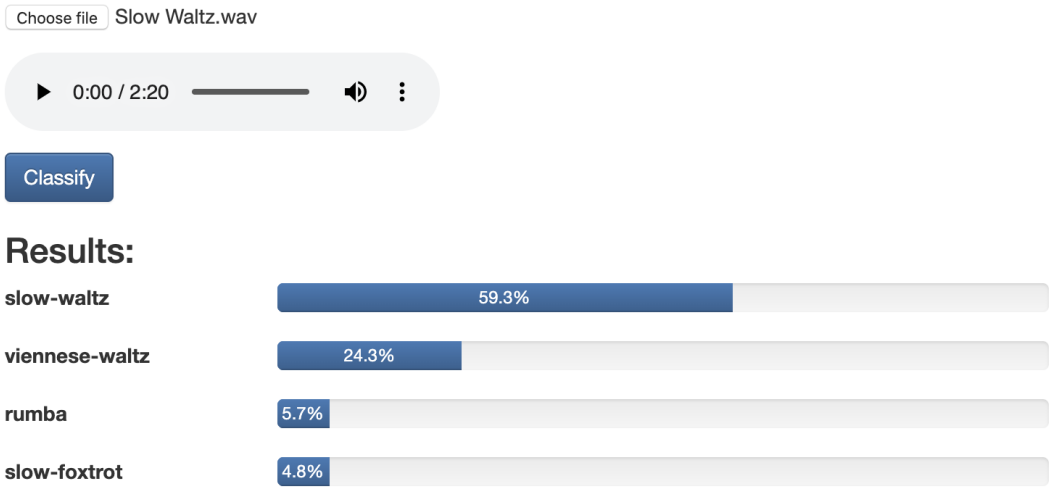

Figure 4.10: Screenshot of the web application that demonstrates our method. The screenshot shows predictions for Slow Waltz. Note, that the Viennese Waltz is second prediction because it is in triple meter. Both Rumba and Slow Foxtrot are similar to Slow Waltz for their slow tempo. The application can be accessed at `http://dance.ironbrain.net`.

4.8.2 Classification Examples on The Test Set

This section will show classification results of several audio recordings from the test set. As audio recordings are classified by independent spectrogram segments each 5.2-seconds long, the resulting probability scores of each segment will be depicted in figures. It enables us to see the classification result for each 5.2-seconds window in given audio recording. The audio is classified in sliding window fashion with given stride. Note, that the aggregated probability score is visually represented by the ratio of areas corresponding to the dance classes.

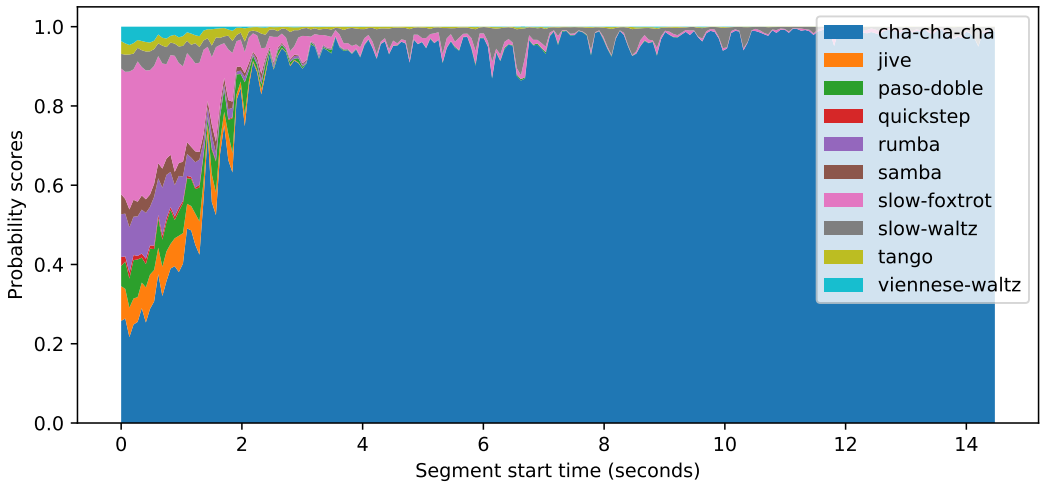

Figure 4.11: Classification of Cha Cha Cha (`https://youtu.be/GHY3Mb4x2Bg` *0:00-0:20*) with fixed intro, that was digitally added before the dance music. The figure shows probability scores for consecutive segments with $stride = 3$.

Audio Recordings with Digitally Added Intro

As mentioned in Section 4.2.2, some of the audio recordings from the test dataset contain fixed few-seconds intro that is digitally added to the recordings beginning. The classification results of recording of Cha Cha Cha, that contains the fixed intro, are shown in Figure 4.11. As the figure illustrates in the beginning, our model is unsurprisingly unable to recognize Cha Cha Cha for 5.2-seconds segments that contain the fixed intro and the correct class Cha Cha Cha is predicted as long as the dance music starts, around time 0:02.

Recordings with Gradual Music Beginning

Classification of some audio recordings has bad results in the beginning of the recording despite the fact that the fixed intro is absent. This is caused by the gradual music beginnings of some dance songs. Even though the beginnings belong to the dance music, the patterns that are required to correctly predict dance class, as beats and timing, are not present in the beginning of the song. Such misclassification of beginning of Tango is shown in Figure 4.12.

As discussed earlier, our results on the Extended Ballroom dataset are slightly lower than those on the test set. We suppose, that this is caused by the reason, that the Extended Ballroom dataset contains 10-seconds song beginnings rather than entire songs. As shown in the Figure 4.12, the 10-seconds beginnings are sometimes challenging to classify.

Another example is shown in Figure 4.13, where both effects, the digitally added intro and gradual beginning of the dance music, are present.

Classification of Similar Dance Classes

Similarities of some dances are noticeable from the model output as shown in Figure 4.14. The similarity of Slow Waltz and Viennese Waltz can be seen in the

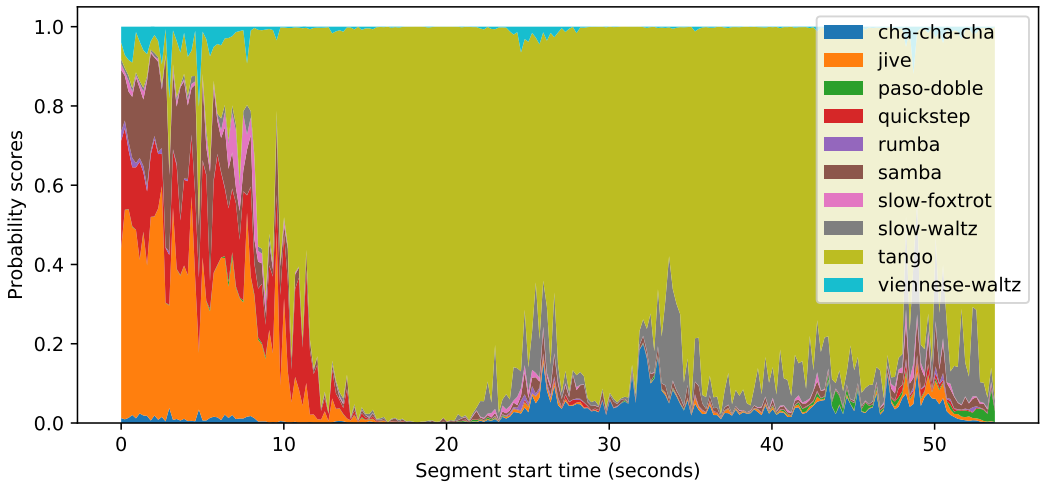

Figure 4.12: Classification of Tango (`https://youtu.be/eC-GLP5WdAg?t=2` *0:02-1:02*) that has song beginning which is challenging to classify. The figure shows probability scores for consecutive segments with $stride = 10$.

figure as well as the slight similarity of latin dances Cha Cha Cha and Rumba. Further, Figure 4.15 shows classification outputs of Slow Waltz recording that is confused with Viennese Waltz for high tempo of the recording. Note, that the classification progress changes in the middle of the recording, where beat pattern changes and the beats start to play slightly louder. The increased beat loudness falsely decrease the probability score of Slow Waltz that is often characteristic by its soft nature.

Audio Events in Dance Music

Our approach of classifying 5.2-seconds segments independently works well for audio recordings with stationary signals. Although, some audio recordings contain musical events as verse change or bridge, that are challenging for the classifier, with 5.2-seconds segments as input, to handle, as shown in Figure 4.16. However, such problematic musical events are unusual for dance music and they are usually only few seconds long. Thus, the presence of the musical events in audio recording does not significantly affect the aggregated results.

4.8.3 Classification Examples on the StarDance Dataset

The audio recordings extracted from StarDance, Czech TV show, contain popular music rather then typical dance music. A prediction of dance styles from popular music is often ambiguous.

Ambiguous Recordings of Popular Music

The classification ambiguity of the StarDance dataset will be illustrated on two audio recordings of Czech Viennese Waltzes from the dataset. While Viennese Waltz can be usually easily recognized for its triple-meter timing, the triple-meter timing is not significant for the two audio recordings of the Viennese Waltz. Figure 4.17 shows ambiguous classification of the two Viennese Waltz recordings

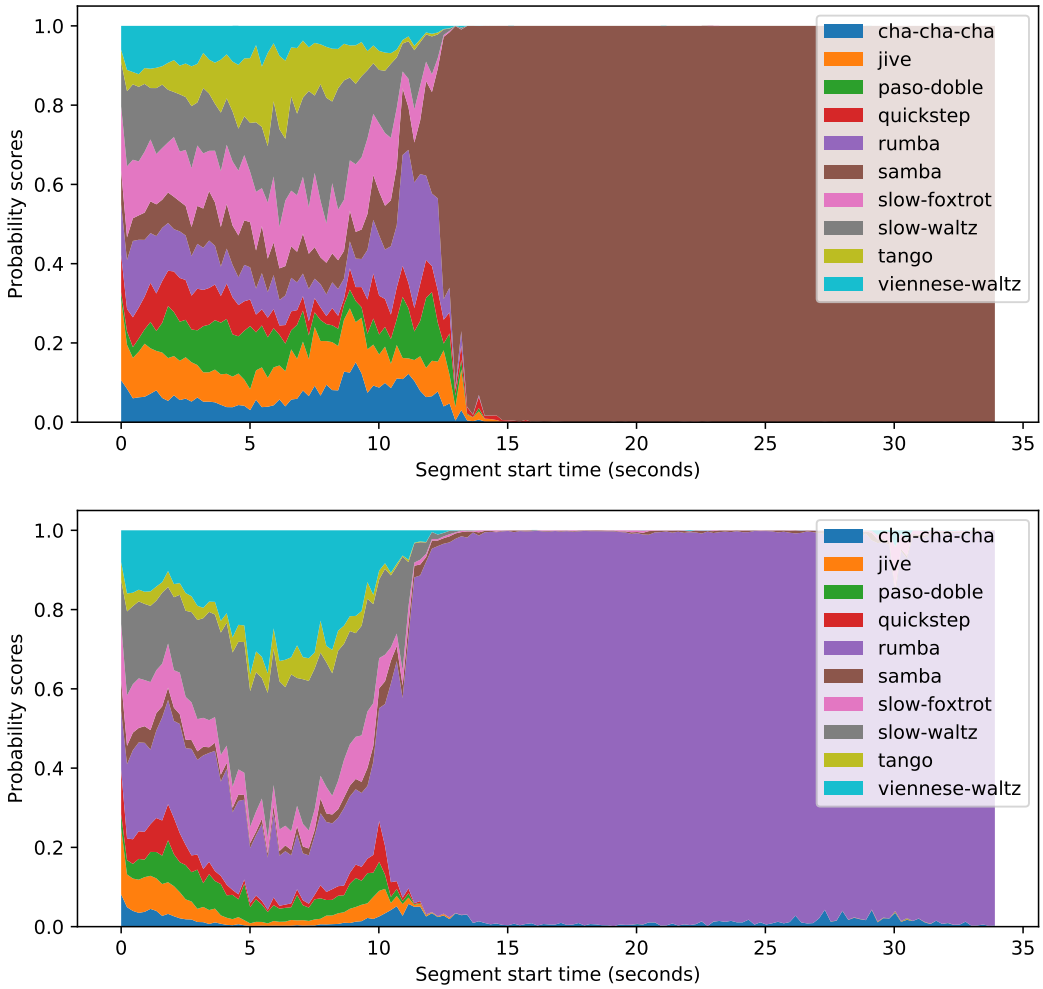

Figure 4.13: Classification of Samba (top, `https://youtu.be/h7hj3KSQ2Ec` *0:00-0:40*) and Rumba (bottom, `https://youtu.be/u7GYcsOkTxw` *0:00-0:40*). Both audio recordings contain both digitally added intro and gradual music beginning. The figures show probability scores for consecutive segments with $stride = 10$.

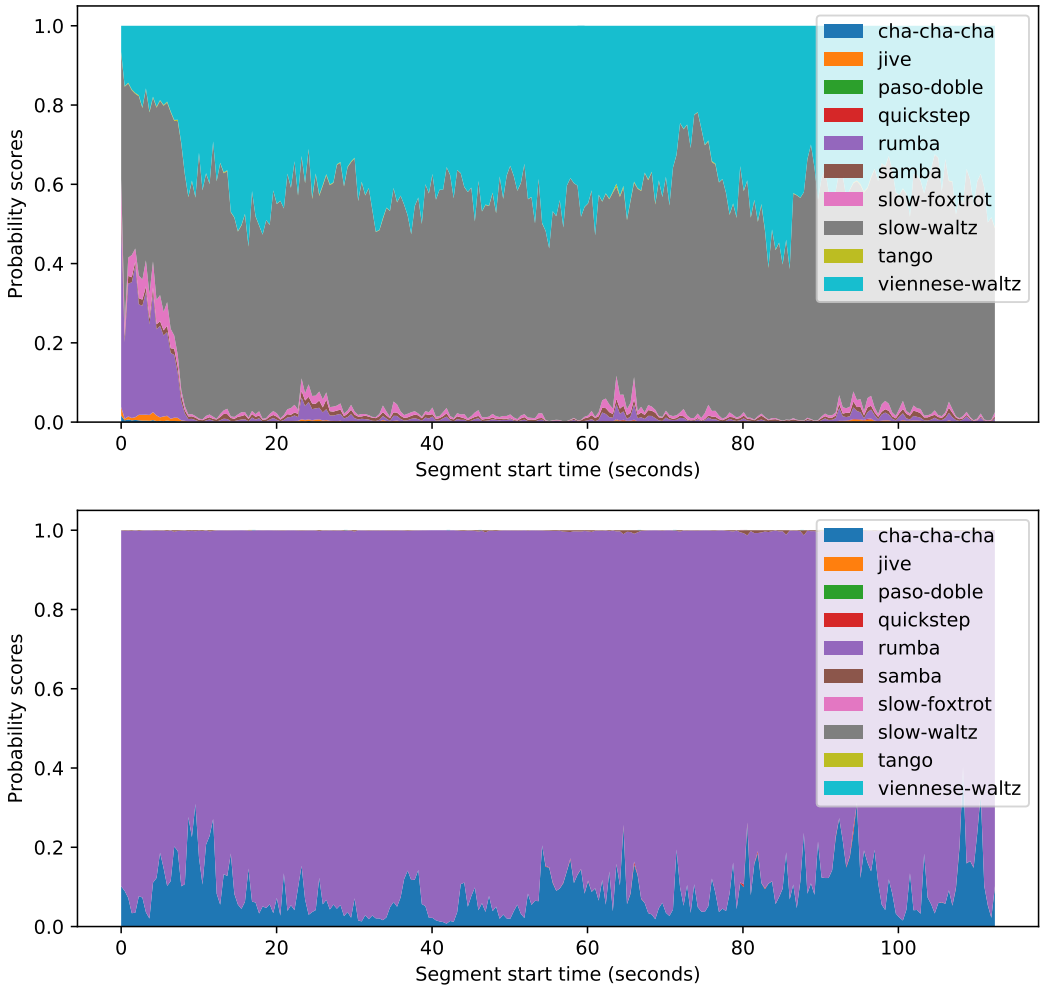

Figure 4.14: Classification of Slow Waltz (top, `https://youtu.be/dVdI1mo_va0` *0:00-2:00*) and Rumba (bottom, `https://youtu.be/3xtV_HnEu48` *0:00-2:00*). Both, the recording of Slow Waltz and the recording of Rumba, are similar to dances Vienesse Waltz and Cha Cha Cha, respectively. The figure shows probability scores for consecutive segments with $stride = 20$.

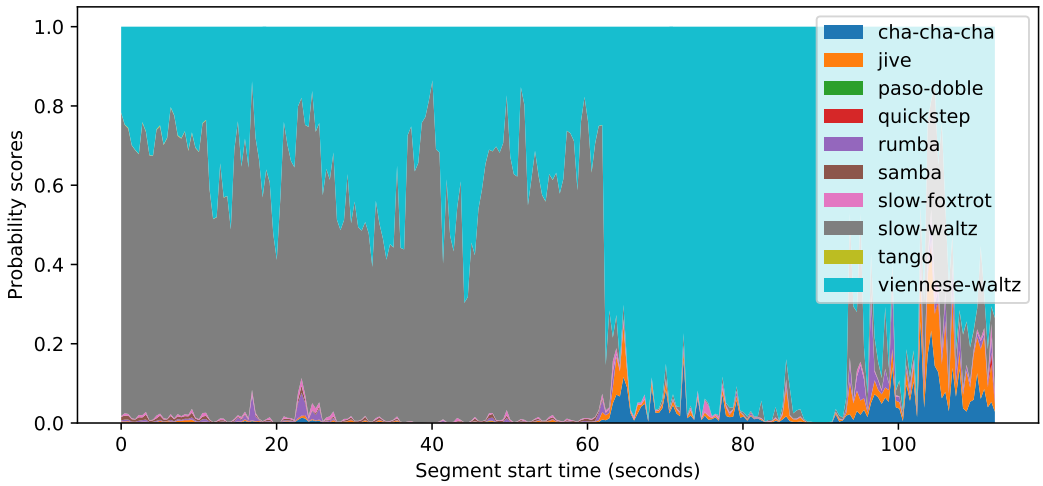

Figure 4.15: Classification of Slow Waltz (`https://youtu.be/j6QCy-19cFs` *0:00-2:00*) that is confused with Viennese Waltz. The change in the middle of the recording classification is triggered by loud beats. The figure shows probability scores for consecutive segments with $stride = 20$.

without significant triple-meter timing. Note, that the probability score of Viennese Waltz increases when the triple meter becomes more remarkable.

Recordings with Czech Music

As shown in Section 4.6, the accuracy on the StarDance dataset is lower than the accuracies of other datasets. To show, that the accuracy is small not by the fact, that the dataset contains Czech songs, we lastly provide example of a Czech song, that is correctly classified as Rumba (Figure 4.18). The song is from Czechoslovak fairy tale movie. Even though the song is not typical dance music, it is classified correctly, because it is played with beats that are typical for Rumba dance. Note, that the audio recording also contains sounds from the StarDance TV show, which cause inaccurate classification in the beginning.

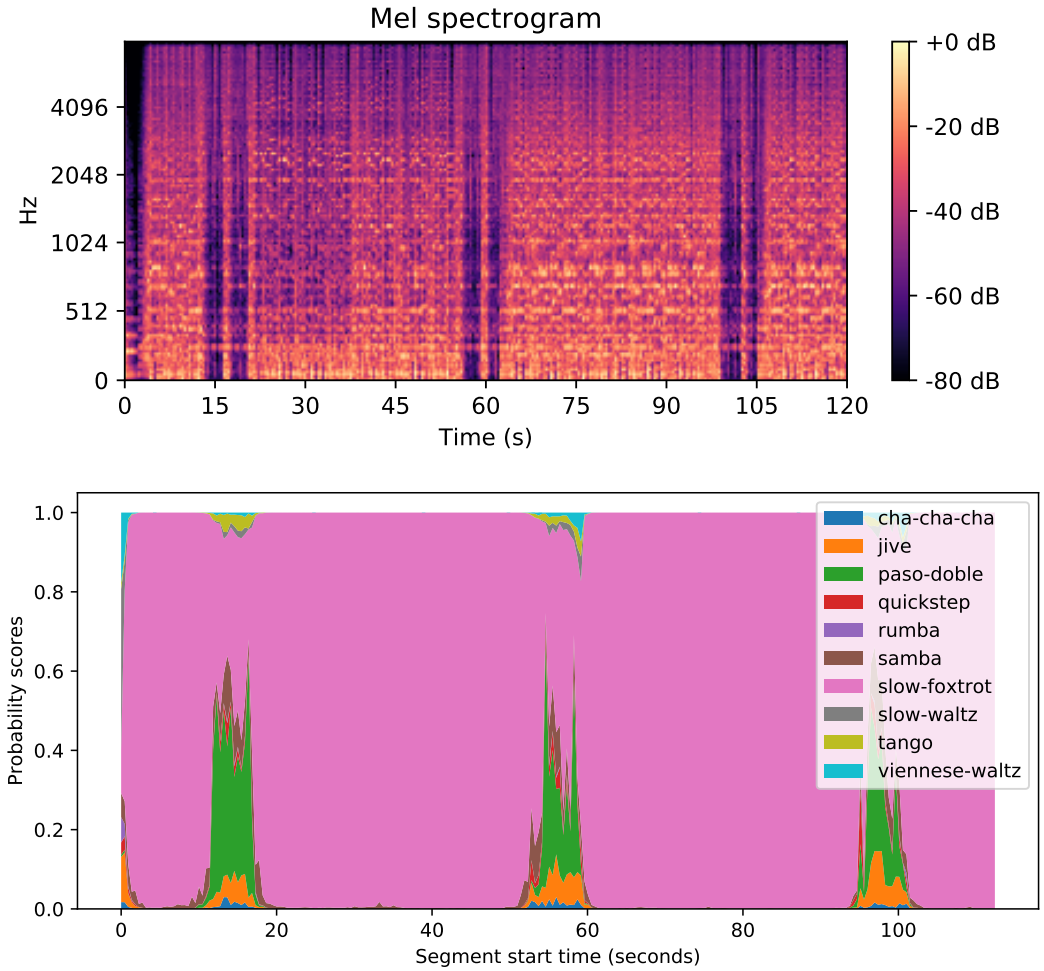

Figure 4.16: Spectrogram (top) and classification (bottom) of audio recording of Slow Foxtrot (`https://youtu.be/ZyYn7Bw3EqA` *0:00-2:00*). As shown in the figure, the recording contains three musical events where music changes and plays with low volume. The bottom plot shows probability scores for consecutive segments with $stride = 20$.

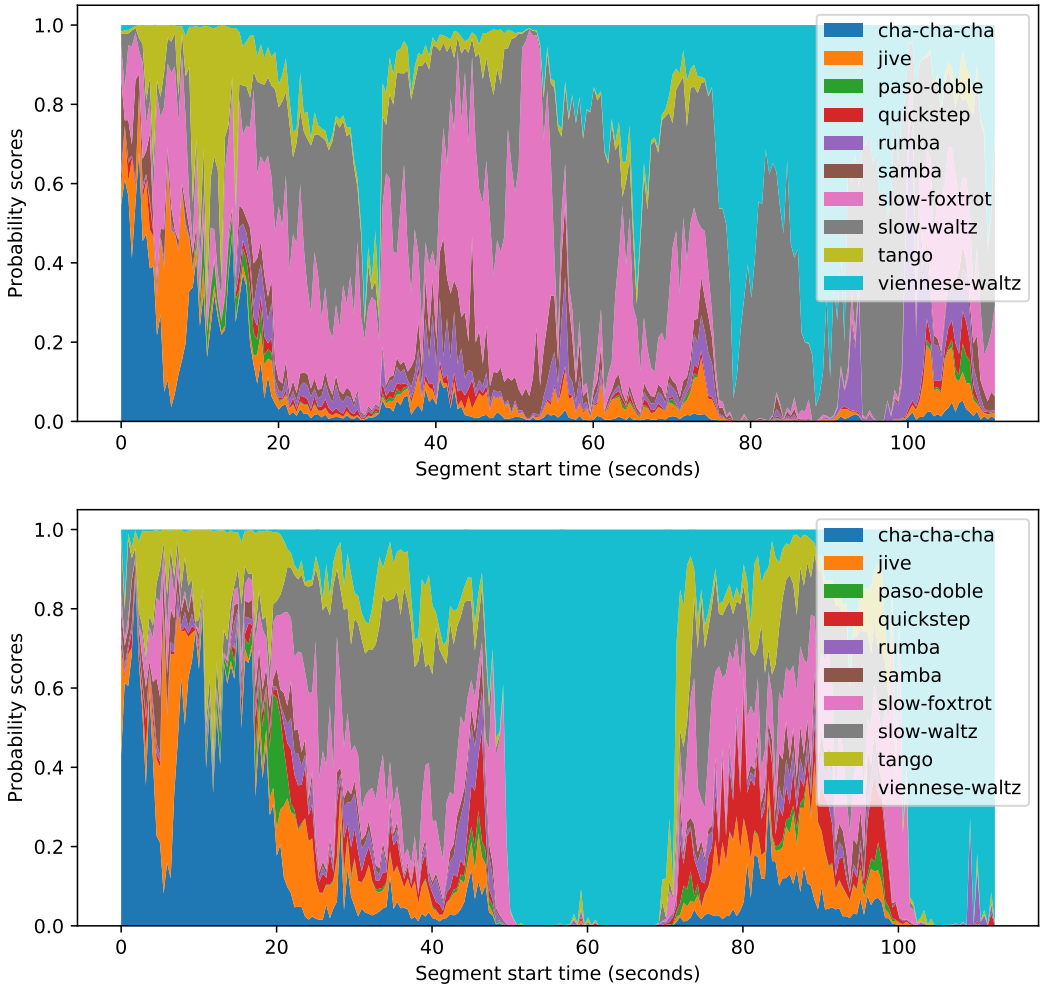

Figure 4.17: Classification of ambiguous recordings of Waltz from StarDance TV show (`https://www.ceskatelevize.cz/porady/12607522764-stardance-x/219544160450001-stardance-x-kdyz-hvezdy-tanci` *Karel Kovy Kovář a Veronika Lišková - Valčík SD 9* and *Gabriela Koukalová a Martin Prágr - Valčík SD 6* resp. both *0:00-2:00*). The recording accompanied triple meter Waltz dance, but the triple-meter timing is not significant in the recordings which causes bad results. The figure shows probability scores for consecutive segments with $stride = 20$.

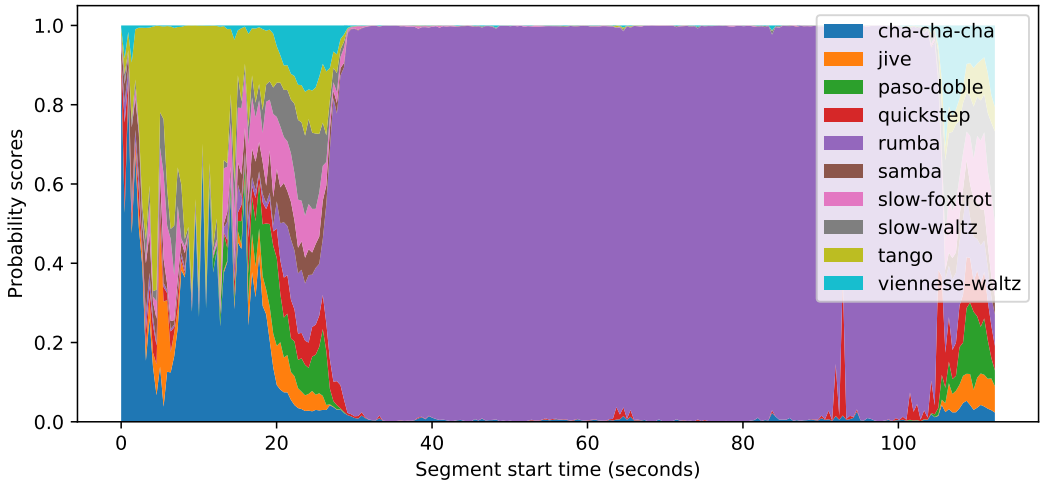

Figure 4.18: Classification of song from Czechoslovak fairy tale movie. The song accompanied Rumba dance in StarDance TV show (`https://www.ceskatelevize.cz/porady/12607522764-stardance-x/219544160450004-stardance-x-kdyz-hvezdy-tanci` *Veronika Khek Kubařová a Dominik Vodička - Rumba SD 2 0:00-2:00*). The audio recording also contains sounds from the show, which cause inaccurate classification in the beginning. The figure shows probability scores for consecutive segments with $stride = 20$.

5. Conclusion

We presented a technique for dance genre classification based on fully convolutional neural networks (FCN). The technique relies on spectrograms that represent a segment of raw audio signal as an image. It was shown, that computer vision approaches can be effectively utilized for predicting dance genres from the spectrograms and we have achieved the best results when relying on pretrained Dense Convolutional Network [Huang et al., 2017], referred as DenseNet, with parameters learned on ImageNet [Deng et al., 2009]. We have shown, that DenseNet slightly outperforms other convolutional neural network architectures as VGG, ResNet and ResNext. We relied on cutting each spectrogram to overlapping segments in sliding window fashion and classified the segments independently. Results has shown, that each 5.2-seconds segment contained sufficient information for correct prediction of a dance class.

We achieved remarkable results of accuracy **96.7%** on our independent test dataset of 60 recordings about four minutes long and **93.9%** accuracy on novel Extended Ballroom dataset [Marchand and Peeters, 2016a] of 4000+ recordings each ten seconds long. We believe, our results are competitive and probably achieves the state-of-the-art in dance genre recognition. We further evaluated our model on various datasets, as audio recordings extracted from dance competitions, and audio recordings extracted from StarDance TV show. The approach was finally evaluated on two datasets with low quality data. Audio recordings in one of the datasets were recorded on a mobile phone and contained noise of surrounding. Second dataset was created synthetically by adding a background noise to each of the recording in the test set with increasing intensity.

Based on the results of our method on low quality data, we extended the training set by adding low quality audio recordings (Section 4.7.1). We have shown, that accuracy on independent low quality data increased significantly, even though the ratio of the low quality data in the training set was as low as 4.9%.

We implemented a baseline algorithm for dance genre classification. The baseline employs digitally extracted features (e.g. chromagram, spectral centroid and Mel-Frequency Cepstral Coefficients) and support vector machines (SVM). The results of our method were compared with the baseline algorithm and with the MASSS classifier proposed by Marchand and Peeters [2016b].

As a demonstration of our method, we presented a web application with functionality to predict dance from audio recording uploaded by user (Section 4.8.1).

With respect to DenseNet architecture, we experimented with a number of layers and we changed the network structure as replacing the first convolutional layer with vertical convolutions (Section 4.5). It was shown, that the original DenseNet with 7×7 convolutions in the first layer slightly outperforms its modifications, where the first layer is replaced with the layer of vertical convolutions. Further, training all model parameters significantly outperformed the approaches, where parameters of first convolutional layers, learned on ImageNet, were frozen and only the parameters of other layers were trained. The training of all model parameters achieved accurate results for both, the DenseNet pretrained with parameters learned on ImageNet and the DenseNet with parameters randomly ini-

tialized.

Suggestions of possible improvements that we keep as a future work, will follow. Our current approach utilizes simple average to aggregate the classification results of spectrogram segments. Future technique could employ more sophisticated aggregation mechanisms as recurrent neural networks.

The softmax output scores do not reflect the posterior probability distribution. We could train the proper network confidence following Franc and Prusa [2019] that would possibly further improve the aggregation results. Moreover, it would allow us to implement a reject option, that would extend the model to abstain from prediction in ambiguous cases or on classes that the model is not trained for.

As our method employs spectrogram as a representation of an audio signal, other methods for the audio signal representation could be investigated [Marchand and Peeters, 2016b]. Moreover, end-to-end techniques could be employed to classify the dance class from the raw signal without utilizing any intermediate representation, as studied by Dieleman and Schrauwen [2014]. The end-to-end approach could work if enough data is available since the neural network has to learn the representation of the raw signal from scratch.

To achieve more accurate results, ensemble of classifiers could be utilized to predict a dance class, similarly as proposed by Silla Jr et al. [2007]. Especially, CNN-based approach for music classification could be ensembled with approach employing manually extracted features. A hand-crafted feature with a clear interpretation, e.g. music tempo, could achieve more accurate results.

Lastly, the web application, that is presented as a demonstration, could be improved. The method could be implemented as a mobile phone application with impressive user interface and with the possibility to classify dance music recorded using a microphone in real-time. The entire classification process could also work locally on the mobile device. Running the classifier offline on limited computational resources would require replacing DenseNet with computation-efficient CNN architecture as SqueezeNet [Iandola et al., 2016], MobileNet [Howard et al., 2017] or ShuffleNet [Zhang et al., 2018].

Bibliography

Hareesh Bahuleyan. Music genre classification using machine learning techniques. *arXiv preprint arXiv:1804.01149*, 2018.

Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference*, 2011.

Elaine Chew, Anja Volk, and Chia-Ying Lee. Dance music classification using inner metric analysis. In *The next wave in computing, optimization, and decision technologies*, pages 355–370. Springer, 2005.

Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. *arXiv preprint arXiv:1606.00298*, 2016.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Yandre MG Costa, LS Oliveira, Alessandro L Koerich, Fabien Gouyon, and JG Martins. Music genre classification using lbp textural features. *Signal Processing*, 92(11):2723–2737, 2012.

Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, 28(4):357–366, 1980.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968. IEEE, 2014.

Simon Dixon, Elias Pampalk, and Gerhard Widmer. Classification of dance music by periodicity patterns. In *ISMIR 2003*. Johns Hopkins University, 2003.

J Stephen Downie. Music information retrieval. *Annual review of information science and technology*, 37(1):295–340, 2003.

Dan Ellis. Chroma feature analysis and synthesis. *Resources of Laboratory for the Recognition and Organization of Speech and Audio-LabROSA*, 2007.

Vojtech Franc and Daniel Prusa. On discriminative learning of prediction uncertainty. In *International Conference on Machine Learning*, pages 1963–1971, 2019.

Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2017.

Fabien Gouyon, François Pachet, Olivier Delerue, et al. On the use of zero-crossing rate for an application of classification of percussive sounds. In *Proceedings of the COST G-6 conference on Digital Audio Effects (DAFX-00), Verona, Italy*, page 26, 2000.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Anssi Klapuri and Manuel Davy. *Signal processing methods for music transcription*. Springer Science & Business Media, 2007.

Stefaan Lippens, Jean-Pierre Martens, and Tom De Mulder. A comparison of human and automatic musical genre classification. In *2004 IEEE international conference on acoustics, speech, and signal processing*, volume 4, pages iv–iv. IEEE, 2004.

Beth Logan et al. Mel frequency cepstral coefficients for music modeling. In *ISMIR*, volume 270, pages 1–11, 2000.

Ugo Marchand and Geoffroy Peeters. The extended ballroom dataset. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2016a.

Ugo Marchand and Geoffroy Peeters. Scale and shift invariant time/frequency representation using auditory statistics: Application to rhythm description. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2016b.

Tomoya Nakai, Naoko Koide-Majima, and Shinji Nishimoto. Encoding and decoding of music-genre representations in the human brain. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 584–589. IEEE, 2018.

Sergio Oramas, Francesco Barbieri, Oriol Nieto, and Xavier Serra. Multimodal deep learning for music genre classification. *Transactions of the International Society for Music Information Retrieval. 2018; 1 (1): 4-21.*, 2018.

Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In *2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6. IEEE, 2016.

Jordi Pons, Oriol Nieto, Matthew Prockup, Erik Schmidt, Andreas Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. *arXiv preprint arXiv:1711.02520*, 2017.

Carlos N Silla Jr, Celso AA Kaestner, and Alessandro L Koerich. Automatic music genre classification using ensemble of classifiers. In *2007 IEEE International Conference on Systems, Man and Cybernetics*, pages 1687–1692. IEEE, 2007.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Chun Pui Tang, Ka Long Chui, Ying Kin Yu, Zhiliang Zeng, and Kin Hong Wong. Music genre classification using a hierarchical long short term memory (lstm) model. In *Third International Workshop on Pattern Recognition*, volume 10828, page 108281B. International Society for Optics and Photonics, 2018.

George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018.

Donglai Zhu, Hengshuai Yao, Bei Jiang, and Peng Yu. Negative log likelihood ratio loss for deep neural network classification. *arXiv preprint arXiv:1804.10690*, 2018.