國立臺灣大學電機資訊學院電機工程學研究所

博士論文

Graduate Institute of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Doctoral dissertation

應用全捲積網路所達成之弱監督音樂音訊及視訊事件偵測

Weakly-supervised Event Detection for Music Audios and Videos Using Fully-convolutional Networks

劉任瑜

Jen-Yu Liu

指導教授：鄭士康教授、楊奕軒博士

Advisor: Prof. Shyh-Kang Jeng and Dr. Yi-Hsuan Yang

中華民國 107 年 6 月

June, 2018

doi:10.6342/NTU201801365

國立臺灣大學博士學位論文
口試委員會審定書

應用全捲積網路所達成之弱監督音樂音訊及視訊事件偵測

Weakly-supervised Event Detection for Music Audios and Videos Using Fully-convolutional Networks

本論文係劉任瑜君 (D02921018) 在國立臺灣大學電機工程學研究所完成之博士學位論文，於民國 107 年 6 月 14 日承下列考試委員審查通過及口試及格，特此證明

口試委員： 鄭士康 (簽名)
(指導教授)

張智星	楊奕軒
深山覺	蘇黎
李宏毅	

系主任、所長 劉志文 (簽名)

致謝

本論文得以完成，首先要感謝兩位指導老師，鄭士康老師與楊奕軒老師。這幾年來你們在專業、做研究的方法與態度上、以及研究結果的呈現上，都給我相當大的指導與幫忙。也要謝謝你們很寬容的接受我對一些問題的看法與研究方法上的固執。謝謝口試委員鄭士康教授、楊奕軒博士、張智星教授、李宏毅教授、蘇黎博士以及深山覺博士撥冗參加我的論文口試，並給予我相當寶貴的建議。這些建議使本論文更加完備，也啟發我對未來研究的一些想法。

有我的家人與朋友在心靈上不間斷的支持，這篇論文才有可能完成。爸爸、媽媽、姐姐、嘉嘉和大腸在我人生的各個階段一直都是我最重要的支柱，尤其在就讀博士班期間更是如此。作研究、投稿和寫論文總會有低潮，有你們的陪伴讓這些苦悶的時刻輕鬆不少，也讓我有繼續堅持下去的動力。沒有你們這篇論文絕對是無法如此順利的完成，謝謝你們！

doi:10.6342/NTU201801365

中文摘要

隨著視訊與音訊串流服務的流行，音樂音訊與視訊是現今最受歡迎的娛樂來源之一。音樂與音樂演奏包含相當豐富的資訊。為了能自動分析這些音訊及視訊以進一步進行檢索或教學，我們會想要使用機器學習來幫助偵測各式音訊及視覺事件。然而，機器學習的方法通常需要相當數量的訓練資料。在音訊及視訊中，標示這些訓練資料並不容易，因為手動標示的過程非常花時間而且乏味。在本論文中，我們探討如何以弱監督的方式，僅使用長片斷層級的標示來訓練偵測模型。我們使用全捲積網路來達到音樂音訊與視訊之事件偵測。首先，使用全捲積網路在時間上偵測音樂音訊事件，如曲風、樂器、情緒等，並且使用樂器演奏資料庫來評估模型的表現。接著，我們將發展一個弱監督的架構來實現視訊中的樂器演奏動作偵測。此學習架構包含兩個輔助模型：聲音偵測模型與物體偵測模型。這兩個輔助模型也只使用長片斷層級的標記來訓練。它們將為動作偵測模型提供監督資訊。我們使用 5400 個經過手動標記的影像畫面來評估此訓練架構的表現。提出之訓練架構在時間與空間上相當大程度地改進了模型表現。

關鍵字：音樂事件偵測、樂器演奏動作偵測、弱監督學習、音樂自動標籤

doi:10.6342/NTU201801365

Abstract

With the growing of audio and video streaming services, music audios and videos are among the most popular sources for entertainment in recent days. There are rich information in music and music playing. In order to automatically analyze these audios and videos for further retrieval or pedagogical purpose, we may want to use machine learning to help with detecting audio and visual events. However, learning-based methods usually require a large amount of training data. In audios and videos, annotating these data are not easy because the process is time-consuming and tedious. In this work, we will see how to train such detection models with only clip-level annotations with weakly-supervised learning. We will use fully-convolutional networks (FCNs) for event detection in music audios and videos. First, we will develop FCNs for temporally detecting music audio events such as genres, instruments, and moods, which will be evaluated on an instrument dataset. Second, we will develop a weakly-supervised framework for detecting instrument-playing actions in videos. The learning framework involves two auxiliary models, a sound model and an object model, which are trained using clip-level annotations only. They will provide supervisions temporally and spatially for the action model. In total 5,400 annotated frames will be used to evaluate the performance of the proposed framework. The proposed framework largely improves the performance temporally and spatially.

Key words: music event detection, instrument-playing action detection, weakly-supervised learning, music auto-tagging

doi:10.6342/NTU201801365

Contents

doi:10.6342/NTU201801365

doi:10.6342/NTU201801365

doi:10.6342/NTU201801365

List of Figures

doi:10.6342/NTU201801365

doi:10.6342/NTU201801365

doi:10.6342/NTU201801365

doi:10.6342/NTU201801365

List of Tables

doi:10.6342/NTU201801365

doi:10.6342/NTU201801365

doi:10.6342/NTU201801365

doi:10.6342/NTU201801365

Chapter 1

Introduction

Music-related audios and videos are among the most popular sources for entertainment. There are rich information in music and music playing. The audio parts are with instrument sounds, vocal sounds, or the pattern of blues, rocks, jazz, etc. Many other properties like tempos, and pitches are often recognized. In a performance recorded in video, we can expect to see instrument-playing actions, the sounds made by the instruments, the expression of the players, the interaction of the players with audience, and so on. Objects such as humans and instruments, as well as actions such as dancing and instrument playing are often included in the scene. All these properties can be considered as events, because they occur in some places during a period of time and are with varying time durations. In order to analyze these audios and videos automatically, it is important to detect these events.

Machine learning provides ways to automatically label these audios and videos. However, learning-based methods also require an amount of data for training the models. Manually labeling the temporal and spatial locations of these events is often tedious and time-consuming.

Let's see what we need to do for annotating a music piece with instrument labels. A difference between annotating an image and annotating a music piece is that we have to take time to listen to the music from the beginning to the end. We cannot skip the middle parts because there might be important information there. As we are listening to the music, we start to annotate the music with instrument labels. However, there are often multiple

doi:10.6342/NTU201801365

instruments playing at the same time and some instruments sound similar, so listening once is often not enough. We have to listen to the same music several times. This is just for one song, and it is very tedious. Imagine how many works we have to do if we want to annotate the instrument objects and instrument-playing actions videos.

Thus the main topic of this dissertation is about how to alleviate the difficulty of labeling annotations by utilizing weakly-supervised learning is the main topic of this work. We will investigate how to use clip-level annotations to train a model that can detect music audio events and instrument-playing actions. In contrast to the difficulty of collecting data annotated with detailed temporal and spatial information, audio and video data with simple clip-level annotations are much easier to acquire from websites such as YouTube and Last.fm as well as publicly available datasets such as YouTube8M and AudioSet.

In this work, methods are proposed to learn the temporal and spatial locations of audio events and actions in audios and videos. For the event detection in music audios, fully-convolutional networks (FCNs) are used as the base models for detection. Two mechanisms are used to further enhance the performance. First, Gaussian filters are used to capture the temporal characteristics of different types of labels, which adapt its width to different labels in the training process. Second, multiple input feature maps with different FFT window sizes are used to capture information from different temporal scales.

Extending from the the method for weakly-supervised music audio event detection, we will see that the instrument-playing action detection can also be done in a weakly-supervised way. For videos, the annotations of actions are even more difficult to collect because they have to be labeled both temporally and spatially. We can observe that instrument playing is among a special type of actions which involves a tool (instruments) and the tool would make sounds (music sounds). This work proposes a way to utilize this property to automatically produce extra supervisions in learning from two auxiliary models.

doi:10.6342/NTU201801365

1.1 Contributions

- A novel model is proposed for weakly-supervised audio event detection by using fully-convolutional networks
- A novel framework is proposed for weakly-supervised instrument-playing action detection by using two auxiliary networks to provide supervisory signals
- A new dataset is annotated and released for evaluating instrument-playing action detection
- Extensive experiments are conducted to evaluate the proposed models
- The codes are made publicly available for both music audios (https://github.com/ciaua/clip2frame) and music videos (https://github.com/ciaua/InstrumentPlayingDetection).

1.2 Event detection in music audios

The problem of event or object localization arises in many areas of information processing. In speech recognition, the temporal correspondence between the spoken and the recognized words is required [3–6]. In visual object recognition, people want to know not only whether an object (e.g., a cat) is present in an image, but also where the object really is within the image [7–11].

The localization problem also exists in music auto-tagging, whose goal is to automatically label a music clip with attributes such as instruments, genres/styles or acoustic properties [12–15]. Although people tend to describe such attributes simply as *tags*, we can also view them as descriptors of music *events*. For example, an instrument tag "guitar" can be thought of as "the presence of guitar." A genre tag such as "Jazz" is usually used to describe a whole piece of music, but it can also be rephrased as "having a Jazz flavor" which describes parts of the piece. From this point of view, a tag should be associated with starting points and ending points in a music piece, if the tag applies to multiple parts of the piece. Localizing the music events is beneficial for many tasks in music information

doi:10.6342/NTU201801365

retrieval. For example, a user would not know that there are Jazz elements in parts of a music piece, if the piece is only globally labeled as being "Pop." In music search, a user may look for short segments of music pieces featuring certain attributes, such as "metal screaming vocal," "a saxophone solo" or "outside improvisation." If several instruments are used in a music piece, localizing the instruments in time gives us a better understanding of the overall structure and organization of the piece.

Music event localization/detection, however, is seldom addressed in the literature, mainly due to the scarcity of labeled data. In speech recognition, localization is achieved by collecting frame-level annotation and training prediction models directly in the frame level [3,4]. In visual object detection, manual annotation of the so-called bounding boxes around the objects of interest is usually needed for training a model for localization [7]. Although music event localization may sound similar to these two tasks, labeling music events by hand is arguably more difficult. For example, while speech data usually have only one active speaker at a time [16], it is typical to have multiple instruments, vocals, or styles in music, rendering it a multi-label problem [17]. As opposed to image, labeling music requires certain level of domain expertise. Localizing events in a time stream is also more labor-intensive and time-consuming. In consequence, it is not surprising that most datasets available for music auto-tagging contain only clip-level annotations [15,17].

In light of these observations, a novel approach is proposed in this work that conquers the scarcity of labeled data for music event localization by using only clip-level annotation of tags in the training phase. In music event detection, the contribution of this work is threefold.

First, this work represents one of the first attempts that systematically investigate the localization problem in music auto-tagging. In addition to building a machine learning model that is able to make both *clip-level* and *frame-level* prediction of tag occurrence, a series of experiments are also presented to gain insights into issues specific to the frame-level prediction task, namely music event detection or localization. In general, "event detection" (or "event localization") is about locating the start and end time of an event, whereas "frame-level tagging/prediction" is about predicting the labels at frame level. In

doi:10.6342/NTU201801365

this work, event detection is achieved by frame-level prediction, and we will use them interchangeably because we can derive the other if we have one.

Second, by extending a convolutional neural network (CNN)-based approach to visual object recognition [10], a fully-convolutional network (FCN) model is developed for predicting and detecting music events in unseen music by training on only clip-level data. Our model is designed to account for the variable duration of music events and the temporal information of music. This is done by adding an *accumulation* layer that uses a Gaussian filter of tag-dependent length for assembling local information, and by feeding multi-scale audio features to the FCN. Different from existing CNN models for musical feature extraction [13, 14], our FCN model can deal with music of arbitrary length. An example of the detection result of the proposed model is shown in Figure 1.1. A song can have different types of properties such as genres, instruments, vocals, and tempos. Furthermore, the properties may only appear at some parts of the song, so we can see in Figure 1.1 that the detection result of different tags do not coincide. The code and the trained model are available online[1].

Third, as no evaluation framework has been proposed for music event localization trained with clip-level annotations, this work is also motivated to propose such a framework to facilitate objective evaluation of the prediction result. In this work, we will see that existing *multi-track* music datasets originally intended for studying musical source separation or transcription problems can be nicely used to evaluate frame-level prediction of instrument tags, and demonstrate such an evaluation by using the music auto-tagging dataset MagnaTagATune [17] and a recently proposed multi-track dataset called MedleyDB [18]. Visualization of some prediction results is also presented, which can be used to subjectively evaluate the performance for both instrument and non-instrument tags.

1.3 Instrument-playing action detection in music videos

With the popularity of social media and online sharing, people are sharing a large amount of videos online every day. These videos often contain human activities, so hu-

[1] https://github.com/ciaua/clip2frame

doi:10.6342/NTU201801365

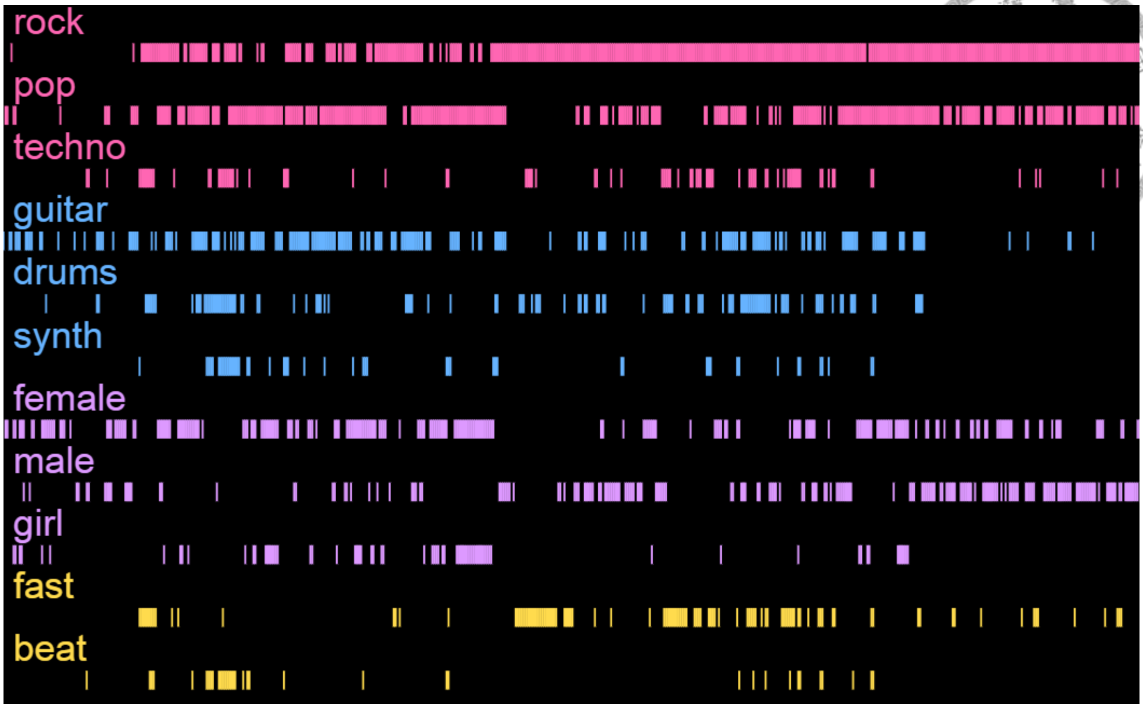


Figure 1.1: An example of the music event predictions.

man actions or movements are informative components in these videos. Therefore, automatically recognizing the types of the actions and locating the actions in videos can help understand and retrieve videos [19]. This task is often called "action detection" [19–26]. For fully-supervised learning approaches, detailed temporal and spatial annotations of actions are usually needed for action detection. However, these annotations are difficult to acquire because the labeling is labor-intensive and time-consuming [19]. In recent years, researchers have proposed various strategies to alleviate this issue [19, 21, 26].

An observation can be made that the objects and sounds in several types of videos might also be used to alleviate this issue. In a large amount of videos, both the objects and sounds signify the key points of the actions. Examples include videos with instrument playing [27], videos with violent content [28], and sport videos [29]. For example, when we hear a guitar solo and see a musician holding a guitar in a video, it is pretty likely that the guitar solo comes from the musician's playing actions. The hitting actions in ball games often contain the acting objects, such as feet, hands, rackets, or bats, as well as the accompanied sounds of hitting. This relationship between actions, objects, and sounds provides an opportunity to infer the appearance of actions from objects and sounds.

doi:10.6342/NTU201801365

Specifically, from the actions in the videos where sounds and objects signify the key points of actions, we can observe the following two common properties:

Action-in-object The spatial location of an action is close to (e.g. at the border or within) the spatial location of objects (e.g., instruments, bats, balls, or weapons).

Action-making-sound A specific type of actions is associated with a specific type of sounds that the actions make.

Action-in-object together with the region of the objects in the scene may give us clues regarding where the actions occur spatially, while action-making-sound together with the temporal activation of the sounds in the video frames may help us temporally locate the actions. In contrast to annotated action data, annotated data of objects and annotated data of sounds are easier to acquire. Therefore, in this work, a method is proposed to train a sound model specifying when the actions occur and train an object model telling us where the objects are. These two auxiliary models act as teachers to inform the action model when and where to pay attention to. We feed only the motion information (dense optical flows in this work) to the action model, so it is forced to learn when and where the actions occur by only motions with the help from the two auxiliary models. An interesting feature of this proposed framework is that it does not need annotated data of actions at all in the training process. This proposed framework is considered as a weakly-supervised learning one, because the model is trained to predict *when* and *where* the playing actions are in videos by using only information regarding *whether* an instrument appears in a video clip in the training phase. The proposed framework is depicted in Fig. 4.2b (Figs. 4.1a, 4.1b, and 4.2a are variants of the proposed framework that will be discussed in Section 4.1.2).

We will focus on the instrument-playing actions in music-related videos in this work. Music is one of the most popular types among online videos (ranked number two according to the study of Cheng *et al.* [30]), and instrument playing is among the most common scenes in these videos. For the audio aspect of instrument playing, automatic detection of instrument sounds has been widely studied in music information retrieval (MIR) [31–35]. It helps people understand the content of the music. However, the visual aspect of instrument playing remains largely unaddressed in literature. In addition to the sounds of instru-

doi:10.6342/NTU201801365

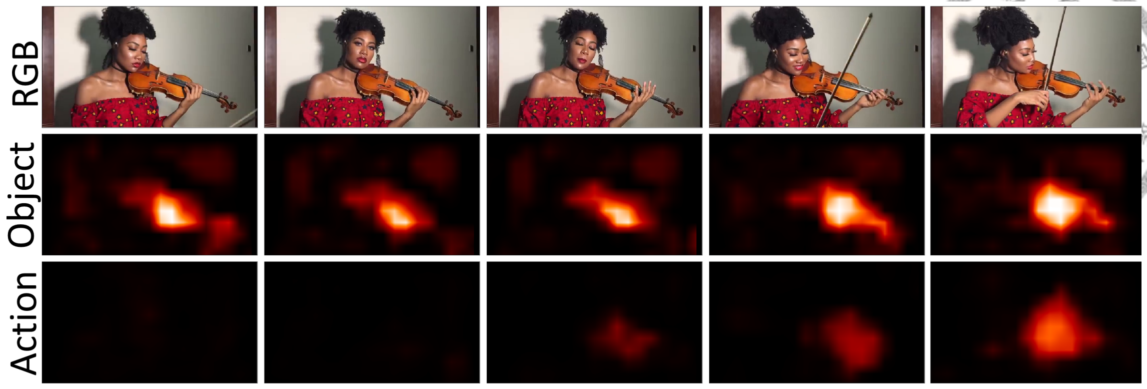


Figure 1.2: An example of the action predictions. They are five consecutive frames with one-second interval.[3]

ments, the visual appearances of instruments and instrument-playing actions also provide us important information about the music-related videos. In order to understand music-related videos, we need to know which instruments are played, when the instruments are played, and where the playing actions occur in the scene.[2] For example, in a video of a piano concert, the pianist may first walk into the scene, sit down, and then start to play the piano. In this case, the piano is not played until the pianist sits down and is ready. We may want to know when the playing begins, the relative position of the piano to the scene, the relative positions of the hands to the piano, etc. There are also attempts to model the audio and visual information jointly for music information retrieval tasks [37,38]. For example, Schindler *et al.* investigated music genre classification by aggregating audio features and visual features together as the input features to a classifier [37]. This approach could improve the input feature of the model, but cannot circumvent the lack of annotated data.

In light of these observations, the goal of this work for music videos is to train a model to automatically pinpoint the instrument-playing actions temporally and spatially in videos with instrument-playing scenes without detailed annotations. In contrast to the abundance of annotated data available for either object recognition (including instruments), such as

[2]And even *how* the instruments are played—the gesture, the playing technique, the expression etc [36]. This is left as a topic of future research.

[3]The RGB snapshots are cropped from an YouTube video (ID: 3hjHJo452dY, uploaded by Zara and Nicola) with Creative Commons license.

doi:10.6342/NTU201801365

ImageNet[4] [39], or sound recognition, such as AudioSet[5], we have no available dataset specifying the location of the playing actions in the scenes. Therefore, we can train the action model by utilizing the two properties mentioned above together with a trained sound model and a trained object model. We use the spatial locations of instrument objects and the temporal locations of the instrument sounds to help the detection of playing actions, but do not join the input features. In this way, we have a more flexible model that can work even if the audio is degraded due to factors such as environmental noises, audio track loss, or audio compression artifacts [40]. We human beings can guess if an instrument is played simply by the action, gesture, and the relative positions of hands or bows to instruments. An example of action detection result is shown in Fig. 1.2. The violinist is not playing initially, and then she gradually raises the bow and starts playing in the final two frames. It shows that the the instruments and the playing actions do not always temporally coincide, and the action model should be able to handle this situation.

In order to investigate weakly-supervised instrument-playing action detection, three aspects are contributed in this dissertation. First, a training framework is proposed to learn the temporal and spatial locations of the actions without detailed annotations by utilizing the object and the sound information. Furthermore, we can utilize the object and sound information to further improve the result after the action model is trained by a simple yet effective method of model fusion. Second, although the proposed method does not require detailed location information in training, for the purpose of evaluation, I manually annotated totally 5,400 frames from 135 videos with detailed locations of instrument-playing actions. Third, comprehensive experiments are conducted to investigate the effects of different components in the framework (Section 4.3). The action patterns the neural network learns for each instrument are analyzed.

[4]http://www.image-net.org/

[5]https://research.google.com/audioset/

doi:10.6342/NTU201801365

1.4 Overview

This dissertation consists five chapters. In this chapter, Chapter 1, the task of event detection in audios and videos is introduced, including some general information and related work. Chapter 2 introduces the background knowledge used throughout this work, including the difference between supervised learning and weakly-supervised learning, the definition of event detection in the context of this work, and the basics of neural networks. Then, the proposed method of conducting weakly-supervised music event detection is detailed in Chapter 3. In Chapter 4, the weakly-supervised music event detection is used to help weakly-supervised instrument-playing action detection in videos. Finally, I conclude this work in Chapter 5.

doi:10.6342/NTU201801365

Chapter 2

Background

In this chapter, we will first survey studies related to this work. Then, concepts used throughout this work will be introduced. The concepts include the definition of event detection, weakly-supervised learning, fully-convolutional networks, and the features used.

2.1 Literature survey

In this section, we will review and discuss the studies related to this work. They are divided into three categories: detection and classification in audios, detection and classification in videos and images, and weakly-supervised learning.

2.1.1 Detection and classification in audios

A considerable amount of work has been made for music auto-tagging, mostly focusing on only clip-level prediction (i.e., whether a tag can be applied to a music piece) [12–15, 41–44]. Although audio features are usually extracted in the frame level, the objective of learning is to make clip-level prediction. In recent years, deep neural network architectures have been found superior to competing machine learning models for music auto-tagging [13, 14, 45]. For example, Dieleman *et al.* have shown the effectiveness of CNN in learning features for music auto-tagging [13]. However, this CNN model can neither deal with music of arbitrary length nor perform frame-level prediction.

doi:10.6342/NTU201801365

Some studies investigate audio auto-tagging with finer granularity. Essid *et al.* apply hierarchical clustering for frame-level instrument recognition with temporal annotations of instrument occurrences [46]. Mandel *et al.* study the tag relationships inside a track and between tags with 10-second clips [47–49]. Frame-level prediction on music auto-tagging was discussed by Wang *et al.* [50]. Parascandolo *et al.* conducts polyphonic sound event detection with recurrent neural networks [51]. The main difference between the proposed method in this work and those in these studies is that the proposed method can predict at a temporal granularity finer than the granularity of the training data.

In recent years, the multimedia and MIR community starts to address the difficulty in collecting training data for frame-level predictions. Kumar *et al.* investigated the problem of audio event detection with SVM and neural networks with weak-labeled data [52]. Schlüter utilized saliency maps to iteratively train a model that can recognize singing voices in the frame level [53]. In this work, we also utilize an FCN model to derive the frame-level instrument sound predictions. However, we will further use the frame-level instrument sound predictions as the training target for the visual action model, not only as the end product itself.

Instrument recognition has been an active research topic in MIR. Essid *et al.* extracted various audio features and applied hierarchical clustering and SVM for instrument recognition [31]. Han *et al.* proposed a CNN structure to recognize the predominant instrument in music [34]. Slizovskaia *et al.* used both audio and visual features as the input and applied CNNs for the task of instrument recognition [38]. The goal of these works is to recognize the instrument sounds with audio or audio-visual information as the input. In contrast, one of the goals in this work is to detect the instrument-playing actions at frame level by the visual cues in a video.

2.1.2 Detection and classification in videos and images

Zhou et al. identified the difficulty in acquiring action annotations for action detection and proposed a way to estimate the temporal and spatial extents of the actions [19]. They proposed a trajectory split-and-merge algorithm to first segment the background and the

doi:10.6342/NTU201801365

foreground moving objects by using dense optical flows, and then they used the segmentation information to derive the temporal and spatial extents of the actions. Then, they used a latent SVM to classify these segmented patches and locate the actions. We share a similar goal to derive the temporal and spatial extents in our proposed framework, but we investigate utilizing two other modalities to estimate the extents, instead of using the dense optical flows.

Oquab *et al.* proposed to use fully-convolutional neural networks (FCNs) to realize weakly-supervised learning for images [10]. By replacing the fully-connected layers in conventional convolutional neural networks (CNNs) [54,55] with fully-convolutional layers, the model produces an output map that indicates the activation values at different locations. We use this method to do spatial weakly-supervised learning for both action detection and object detection in this work.

There have been several studies on weakly-supervised object detection or segmentation. Hartmann *et al.* used support vector machine (SVM) [56] for weakly-supervised object segmentation in videos [57]. Liu *et al.* used a nearest neighbor-based method to perform weakly-supervised object segmentation in videos [58]. Prest *et al.* used motion cues to produce candidates of temporal tubes that locate a moving object and trained the object detector with a subset of the tubes [59].

Bojanowski [21] and Huang *et al.* [26] tackled the problem of weakly-supervised action detection. In their study, they only knew the sequence of actions and they had to align the actions with the frames in a video clip. They proposed different ways to align the action sequence. Our work is different from theirs in two ways. First, we use auxiliary sound and object models to learn to assign labels to video frames, instead of based on the sequence of labels assigned by human. Second, they only attempt to predict the labels temporally but not spatially.

Simonyan *et al.* proposed a two-stream framework for action detection by using an object stream and an action stream [22]. They experimented with fusing the two streams either by averaging the output scores of the two models or by using SVM to do the final classification. Feichtenhofer *et al.* extended Simonyan's work by using different ways of

doi:10.6342/NTU201801365

model fusion [25], and Ng *et al.* extended Simonyan's work by incorporating information across longer period of time through temporal pooling and LSTM [23]. Our method also contains multiple streams. However, we use FCNs for all the three streams instead of the conventional CNNs because we want not only to classify the videos but also to locate the instruments, the actions, and the sounds. Furthermore, the models are fused only after they are separately trained.

The proposed method for action detection in this work is also related to supervision transfer introduced by Gupta *et al.* [60]. Given two learning tasks where task 1 has large annotated data while task 2 does not, Gupta *et al.* proposed to use the output of a middle layer in the well-trained network in task 1 to provide supervision to a middle layer of the network in task 2. In this way, the supervision is transferred. In this work, we also want to seek for more supervisions to the instrument-playing actions from two other modalities, but we provide the supervisions directly in the output layers by the physical relationships of the three modalities that are indicated by the two observations stated in Section 1.3. The temporal and spatial supervisions are also exploited in addition to the instance-level label supervision in this work.

2.1.3 Weakly-supervised learning

In recent years, unsupervised learning, weakly-supervised, and semi-supervised learning have received lots of attentions in video processing [61–64]. This trend is partly due to the lack of supervisory signals in videos, but it is also because the multi-modal nature of videos and the temporal continuity of videos provide a good environment for learning feature representations by the dependencies between modalities or between frames without external supervisions. For example, Aytar *et al.* [61] and Arandjelović *et al.* [63] proposed to match the audio and visual information to unsupervisedly learn features from a large amount of videos and use only a few labeled data for training a classifier based on the learned features. Aytar *et al.* [64] further included text in addition to the audio and visual information for feature learning. In contrast to the aforementioned multi-modal approaches, Canziani *et al.* [62] proposed a CortexNet framework to learn features by

doi:10.6342/NTU201801365

matching neighboring frames in videos. Similar to these works, our proposed framework also represents an attempt to increase supervisory signals by utilizing multiple modalities of videos for a challenging action detection task.

Labeling the bounding boxes of objects in an image requires more labors than annotating the presence of objects does. Therefore, weakly-supervised approach for visual object localization with only image-level annotations has attracted increasing attentions in recent years [8–11]. Among the prior arts, our approach is closest to the model proposed by Oquab *et al.* [10], a multi-instance learning variant and also based on CNN. A key idea proposed in this work is to use the so-called *full convolutions*, so that the model can process images of arbitrary size. In this way, they can resize an input image arbitrarily in a multi-scale manner to locate a visual object. Being inspired by this approach, our model has two distinct features. First, we adopt a different way to achieve multi-scale learning for audios, as music cannot be easily "resized" as images. Second, we use a dedicated layer to deal with the temporal dimension in music, which is absent in images.

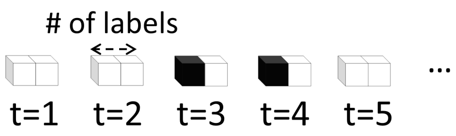
Visual event or action detection in videos, which also have a temporal dimension, has also been studied [22, 65, 66]. Similar to music event detection, this problem requires weakly-supervised learning because the annotation is at the video clip level. However, little work, if any, has been proposed to address localization problem for visual events in videos.

2.2 Event detection as a multi-label classification problem

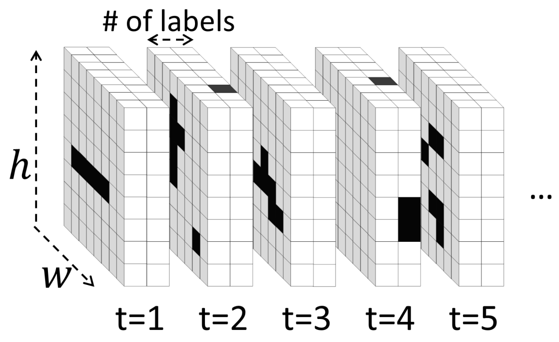
This work tackles the problems of event detection in both audios and videos. The detection in audios involves locating the events in time, while the detection in videos involves locating the events in space and time. Both of the two cases can be seen as multi-label classification problems.

First let's consider the case of audios. In audio event detection, we want to label each temporal point with the properties we are interested. The desired output is in the form

doi:10.6342/NTU201801365



(a) Audio



(b) Video

Figure 2.1: Desired output of event detection in audios and videos

$A = [A_1, A_2, ..., A_T]$, where $A_t = [a_1, a_2, ..., a_K]^T$ is a column vector of K elements, $a_k \in 0, 1$, and K is the number of the labels in the entire dataset. An illustration is shown in Figure 2.1a. A can be seen as a $K \times T$ matrix.

In video event detection, we want to label each pixel at each temporal point with the properties we are interested. Therefore, the desired output is in the form $V = [V_1, V_2, ..., V_T]$, where $V_t = [v_1, v_2, ..., v_K]$ is a $K \times H \times W$ tensor, each v_k is a $H \times W$ matrix representing the spatial information, K is the number of the labels in the entire dataset, and H and W are the height and width of the image frame. An illustration is shown in Figure 2.1a. V can be seen as a $K \times T \times H \times W$ tensor.

For the event detection in either audios or videos, there are no constraints on how many labels a spatial or temporal location can have, so they are multi-label classification problems [67].

A more common tasks than muti-label classification problems are multi-class classification problems. In a multi-class classification problem, it is assumed that there is only one correct answer at a given unit. For example, in image recognition, it is assumed that each image contains only one prominent object [55]. In semantic segmentation, it is also assumed that there is only one class at each pixel [68]. In genre classification in music, it is also often assumed that there is only one genre in one music piece [69].

In contrast, in multi-label classification problems, we could assign 0, 1, or more correct labels at a given unit. In music auto-tagging, a music piece can have multiple properties at the same time. For example, a Rock song can be labeled with ‘Rock,’ ‘Male,’ ‘Female,’

doi:10.6342/NTU201801365

‘Happy,’ ‘Guitar,’ ‘Drum,’ and ‘Bass’ at the same time.

There are some implications due to this difference in practice. In the context of supervised learning, assume we have a vector $x = [x_1, x_2, ..., x_n]$ which is the output of the final layer of a network at a given temporal/spatial location for n labels and a vector $y = [y_1, y_2, ..., y_n]$ which is the corresponding annotation. In a multi-class classification problem, the nonlinear function applied to the output of the final layer of a network is usually a softmax function [55]:

$$Softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)}, \tag{2.1}$$

and the corresponding loss function is a multi-class cross entropy:

$$\sum_j -y_j \cdot \log x_j \tag{2.2}$$

.

In contrast, in a multi-label classification problem, the nonlinear function applied to the output of the final layer of a network is usually a sigmoid function [1]:

$$Sigmoid(x_i) = \frac{1}{1 + exp(-x_i)}, \tag{2.3}$$

and the corresponding loss function is a binary cross entropy:

$$\frac{1}{n}\sum_j -(y_j \cdot \log(x_j) + (1 - y_j) \cdot \log(1 - x_j)) \tag{2.4}$$

.

Furthermore, a threshold is needed in order to turn the real-valued network output into a yes/no binary output in a multi-label classification problem, while a threshold is not required in a multi-class classification problem.

doi:10.6342/NTU201801365

2.3 Weakly-supervised learning

In contrast to the most common supervised learning in classification problems, weakly-supervised learning utilize data that have less or weaker annotations. Zhou recognizes different types of weakly-supervised learning [70], including incomplete supervision, inexact supervision, and inaccurate supervision. What these have in common is that they all have some sort of information loss in the supervisions.

In incomplete supervisions, some data are labeled but some are not labeled. In inaccurate supervisions, the annotations may contain errors [70]. One major source of inaccurate supervisions is the web-crawled data [71]. In inexact supervisions, some supervisions are provided, but is not as exact as desired [70]. Inexact supervisions often appear in detection tasks such as semantic segmentation in images [72], action detection in videos [26], or sound event detection in audios [52]. These tasks are sometimes called multiple instance learning (MIL) [73]. For example, in an MIL situation, we are only told by the supervision that an image contains a dog, but we are not told where the dog and the dog could occupy only a small portion of the image. Supervisions are given, but they are not as exact as we desire.

In the existing works, 'weakly-supervised learning' usually refers to the inexact supervisions [1, 10, 74, 75]. This type of weakly-supervised learning is also what will be discussed in this work.

For either weakly-supervised learning or supervised learning, we may consider a dataset $D = \{(X_1, y_1), (X_2, y_2), ..., (X_M, y_M)\}$ [70], where X_i is an input sample and y_i is the corresponding annotation. X_i could represent different things in supervised learning and weakly-supervised learning.

Assume that we want to know whether a song contains guitar playing and where they are. In this case, X_i is a music audio signal or feature of the shape $K \times T$, where K is the dimension of the feature and T is the temporal unit, and the desired output is z_i with size T, indicating the presence of guitar sounds at each temporal unit.

In supervised learning, we have fully-annotated data, that is, we have annotations with the same size as our desired output. Therefore, y_i has size T as z_i.

doi:10.6342/NTU201801365

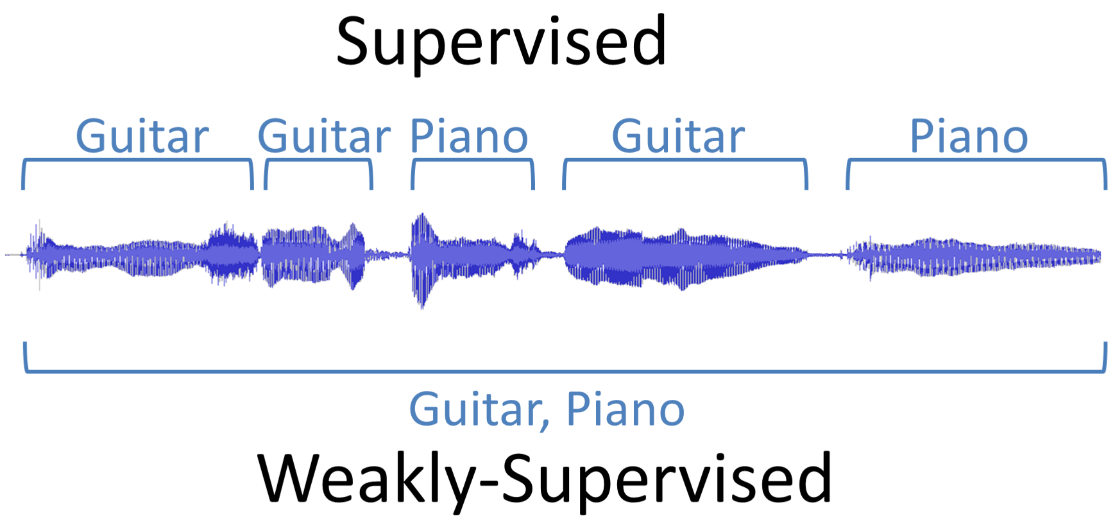


Figure 2.2: Supervised learning VS. weakly-supervised learning

In weakly-supervised learning, we do not have fully-annotated data. For example, a common situation is that we only know whether a song contain guitar sounds, but do not know where the guitar sounds are in the song. In other words, our annotation y_i is simply a scalar with size 1 indicating whether the clip contains guitar or not. The goal, however, is still to derive z_i with size T.

The illustration is shown in Figure 2.2.

2.4 Fully-convolutional networks

In this section, the fully-convolutional networks (FCNs) will be introduced. We will start from deep neural networks (DNNs).

Deep neural networks (DNNs) signify the revival of the neural networks [76]. A DNN stacks several layers of neurons. One layer of DNN is usually composed of a linear transformation, $f(x) = Wx + b$, with weight W and bias b, followed by a nonlinearity function, g, such as sigmoid function, hyperbolic tangent function, or Rectified Linear Unit (ReLU) [77]. Given a input x to a layer, the output is in the form $g(Wx + b)$. MLP will stack 2 or 3 such layers.

Convolutional neural networks (CNNs) can be seen as a generalization of DNN. It

doi:10.6342/NTU201801365

utilizes the convolution operation at each location. Conventional CNNs often use several fully-connected layers on top of the convolutional layers. These final fully-connected layers can be seen as the final classifier. Real breakthroughs in the performance of various tasks start from the CNN architecture, AlexNet, proposed by Krizhevsky *et al* for the Imagenet task. Notable architectures include AlexNet [55] and VGG [78]. Max-pooling is often used after convolution layers as a means to achieve a certain degree of invariance [78].

Fully-convolutional networks (FCNs) are similar to CNNs, but they use only convolution layers without final fully-connected layers. This design has the benefit that it can naturally handle arbitrary size of inputs. Furthermore, the outputs also preserve the location information. A conventional CNN can often be modified into an FCN by replacing final fully-connected layers with convolution layers [22, 79].

2.5 Audio and visual features used

2.5.1 Audio features

For audios, we mainly use mel-scaled spectrogram (mel-spectrogram). The feature is extracted as follows. First, the short-term Fourier transform is applied to a raw audio signal (waveform) and a complex-valued time-frequency matrix is derived. The spectrogram is derived from the time-frequency matrix by taking the absolute value of each complex-valued term. The spectrogram is then transformed into a mel-scaled spectrogram by merging frequency bins according to the mel scale. Finally, a compression [13] is applied to the mel-scaled spectrogram by $log(1 + s * x)$, where s is a chosen positive compression factor and x is the mel-scaled spectrogram. This compression operation is a common practice in audio processing

The specific setting of feature extraction will be described in each section.

doi:10.6342/NTU201801365

2.5.2 Visual features

For images, RGB images are used with normalization. For capturing the movements, dense optical flows are used [22, 80]. The specific usage of these visual features in this work is introduced in Section 4.2.2.

doi:10.6342/NTU201801365

Chapter 3

Weakly-supervised music event detection

In this chapter, we will see how to use fully-convolutional networks (FCNs) to achieve event localization in audios. The chapter is organized as follows. The proposed method is presented in Section 3.1. The datasets used in our evaluation are described in Section 3.2 and the evaluation result is presented in Sections 3.3 and 3.4. An extension of the proposed model to audio event detection is introduced in Sectoin 3.5. Section 3.6 summarizes this chapter. Note that most content in this chapter has been published in the paper [1].

3.1 Proposed method

Our goal is to develop a model that fulfills the following requirements: 1) can give accurate frame-level prediction in the test phase, given only clip-level annotations in the training phase; 2) can handle music pieces of arbitrary length; 3) can capture the temporal characteristics of music events; and 4) can employ multi-scale features as the input.

We propose to achieve this by using a CNN-based architecture for its well-demonstrated effectiveness in various classification tasks in speech recognition [6], computer vision [55], and also music information retrieval [13]. Moreover, neural networks are modular so it is easy for us to add or change components to attain different goals.

The first requirement is met by assuming that the clip-level prediction is aggregated

doi:10.6342/NTU201801365

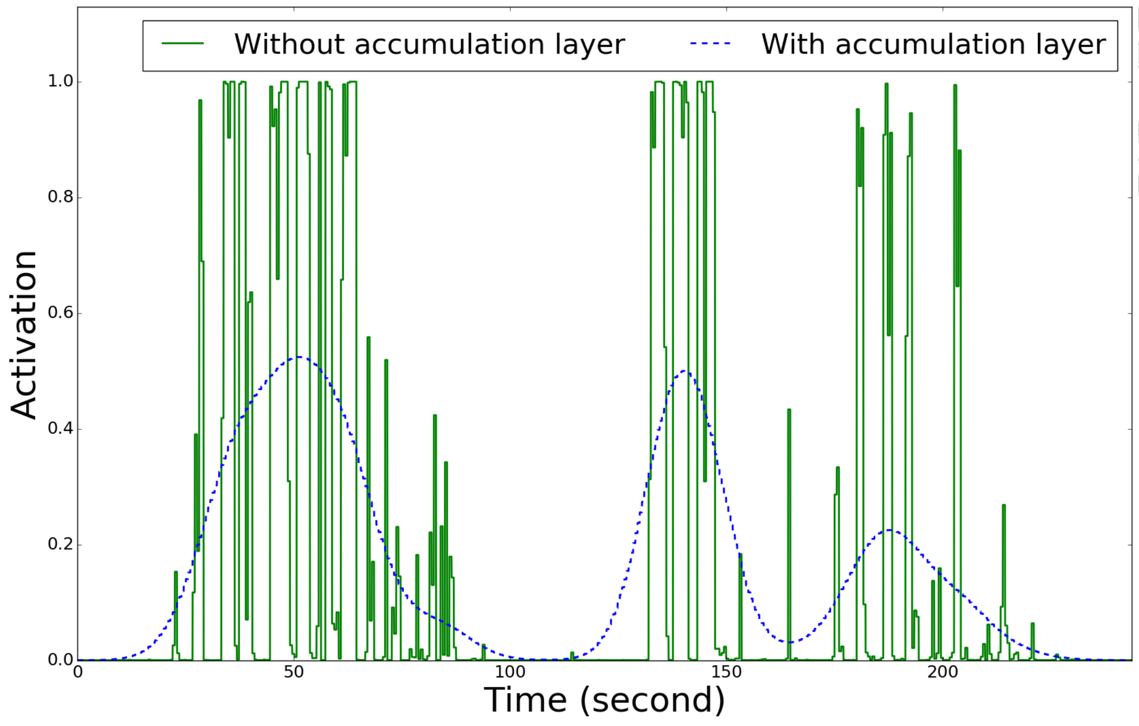


Figure 3.1: Expected effect of the accumulation layer.

from *segment-level* predictions, and that accordingly frame-level predictions can be recovered from the segment-level predictions. Here, a segment is considered as a continuous subset of a music piece that spans a number of frames.

The ability to handle music pieces of arbitrary length is realized by implementing the CNN as a *fully convolutional network* (FCN), where all the layers are convolutional [10]. That is to say, the so-called *fully connected* layers commonly used in CNNs are replaced by convolution layers with window size being 1. This structure has been used in the computer vision community to process images of arbitrary size [10, 81], but it has not been used in music audio before.

The third requirement is important because the *noticeable duration* of music events may be event-dependent. For example, if a guitar sound only lasts for a very short period of time, it can be either inaudible to human ears or too short to be considered as being present. Henceforth, we may not want to annotate the music piece with the tag "guitar." Furthermore, when there is a music event, its typical duration also depends on the nature of the event. For example, instrumental events may be shorter than genre-related events.

To account for these considerations, it is proposed in this work to add an *accumulation*

doi:10.6342/NTU201801365

layer before the output layer. The role of the accumulation layer is to summarize the predictions made from the previous layer over time, as illustrated in Figure 3.1. While the typical CNN architecture can already capture the temporal context of the input features by using convolutions, the accumulation layer is needed to capture the contextual information of music events at the decision-level.

In this work, the accumulation layer is realized by a Gaussian filter. The shape of the Gaussian is controlled by its standard deviation σ, which is a parameter to be learned for different events in the training phase. It is expected that the model will learn different σs for different events. An alternative method is to use the so-called *recurrent* layers for decision-level accumulation. Although the recurrent layers can capture contextual information in a more complex way, this complexity comes at a price of increased computational cost. It is therefore left as a possible future work.

Finally, extracting audio features of different scales is also important in music audio processing, as demonstrated by Dieleman *et al.* [14] in the context of music auto-tagging. We attain this requirement by using multiple stacks of convolution layers to deal with audio features of different resolutions, and later on combining them by concatenation. Different resolutions of audio features are obtained by using the log-scale mel-spectrogram with different window sizes in short-time Fourier Transform (STFT).

The general structure of the proposed network is depicted in Figure 3.2. According to our design, the proposed model can make both clip-level and frame-level predictions. The details are provided in what follows.

3.1.1 Clip-level Prediction

As Figure 3.2 shows, the input layer consists of log mel-spectrogram with N different STFT window sizes [14]. As these are frame-level features computed over time, we refer to a log mel-spectrogram computed with a given window size as a *feature map*. Then, each feature map is processed by its own stack of alternating convolutions and max poolings, referred to altogether as the *early convolution* layers. Max pooling is applied only along the time axis (but not the frequency axis) for every S frames, which is referred to as the

doi:10.6342/NTU201801365

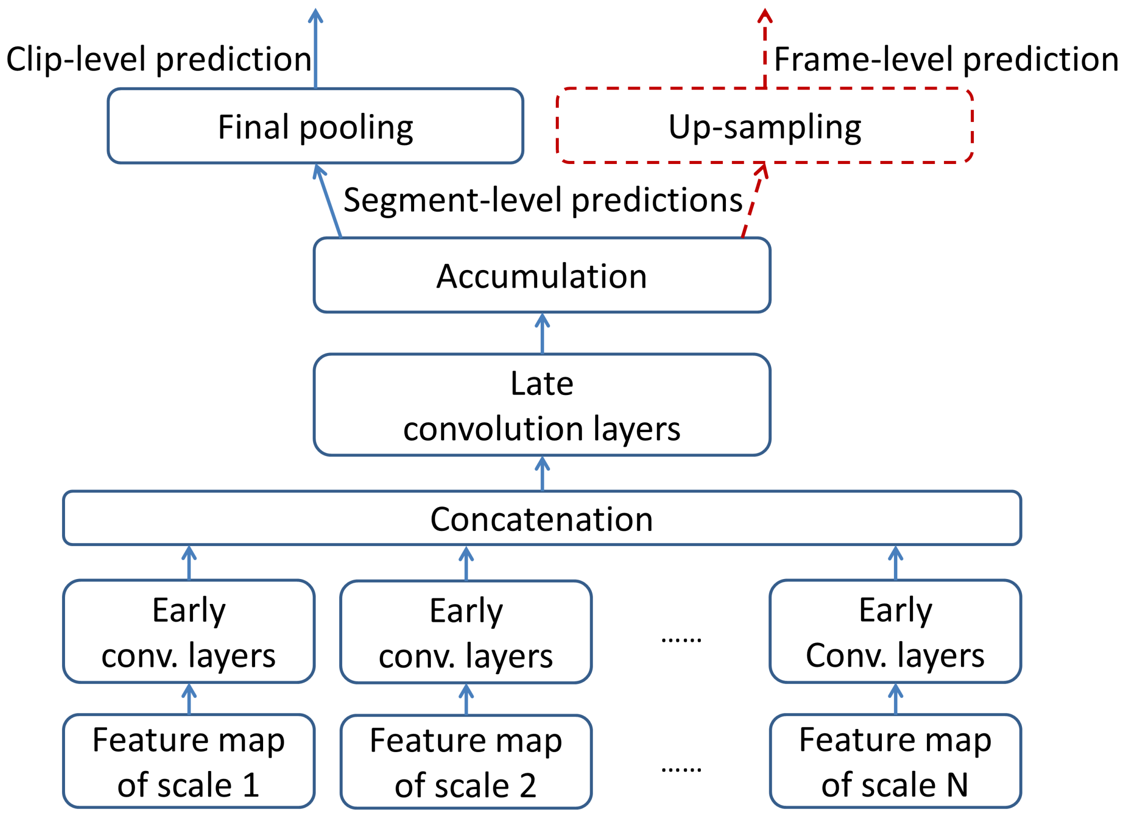


Figure 3.2: The proposed FCN architecture. Given a trained network for clip-level prediction (the blue one), we replace the final pooling layer by an up-sampling layer for frame-level prediction (the red component).

stride size. All stacks of early convolution layers in the proposed FCN structure have the same number of layers. The outputs of the last early convolution layers are concatenated together into one feature map, which aggregates information from different scales. This united feature map is fed into another stack of convolutions, referred to as the *late convolution* layers. The late convolution layers use only convolutions with no strides (i.e., $S = 1$) and no max poolings. They provide the function of fully connected layers commonly used in conventional CNNs and have the benefit of efficiently scanning through a time sequence of arbitrary length. The output of the last late convolution layer is then processed by a Gaussian filter across time to create the *segment-level* predictions. A segment would contain multiple frames, because of the previous max pooling layers. If the σ in the Gaussian is trainable in the training phase, we call it "adaptive Gaussian filter"; otherwise, we call it "fixed Gaussian filter." Finally, the segment-level predictions are pooled over time with the *final pooling* layer to become one clip-level prediction.

doi:10.6342/NTU201801365

To facilitate the later discussion, we define the notion of *total stride size* as the product of all the stride sizes in a stack of early convolution layers. It can be shown that the number of frames covered by a segment in our model equals the total stride size.

3.1.2 Frame-level model

Given a trained network for clip-level prediction, only the last layer is modified to make it possible to make frame-level predictions, as illustrated also in Figure 3.2. Therefore, as there are no trainable parameters in the last layers, the parameters of all layers are shared among the networks for making clip- and frame-level predictions.

Specifically, the aforementioned final pooling layer is now replaced by a *up-sampling* layer to recover frame-level predictions from segment-level predictions. Two methods of up-sampling are investigated in this paper. The first method is *upscaling*, which simply repeats the segment-level predictions locally such that the final output has the same number of frames as the input. For example, suppose there are two early convolution layers and each of them has stride size 2. The total stride size is then 4. Accordingly, the output, say [0, 1, 0], of the accumulation layer has to be repeated 4 times, that is, [0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0].

The second method is *patching*, which has been used in related work in computer vision [9, 82, 83]. Assuming that the total stride size is 4, two temporally neighboring points in the segment-level result can be seen as being 4-frame apart. To fill the values in between, we can simply shift the audio input 4 times and feed each of them to the FCN. This might be more accurate than the upscaling method, but obviously it is computationally much more expensive.

Thresholding

An important issue not depicted in Figure 3.2 is the necessity to apply a thresholding on the frame-level predictions to get binary decisions. This is because the output of FCNs lies in the range between 0 and 1 but we may need binary decisions to evaluate the prediction accuracy. As the optimal threshold values may be tag-dependent, the values

doi:10.6342/NTU201801365

are empirically tuned in the following brute-force way by using validation data (see Section 4.3). First, we divide the output range [0, 1] into K threshold candidates, namely $\{0, 1/K, 2/K, ..., (K-1)/K\}$. For each music event, the F1 scores (i.e., the harmonic mean of precision and recall rates) of frame-level prediction scores are computed with respect to all the threshold candidates, and the candidate with the highest F1 score is selected.

3.2 Data: MagnaTagATune and MedleyDB

We will use two datasets: MagnaTagATune for clip-level training, and MedleyDB for frame-level evaluation.

MagnaTagATune is a music dataset containing clip-level tag annotations [17]. Through letting human subjects play a game, the clips in the dataset were annotated by collecting the human evaluations of music tags generated by an algorithm. It includes 25,863 29-second clips, annotated with 188 tags. The tags include instruments such as "guitar" and "flute," genres such as "Jazz" and "new age," tempo such as "slow" and "fast," and acoustic properties such as "silence" and "noise."

MedleyDB is a multi-track instrument dataset [18]. It has instrument annotations and the corresponding F0 annotations at the frame level, so it can be used for research of melody extraction and source separation. It includes 122 multi-track clips and totally 81 instrument tags. The timestamps of instrument activations are provided in terms of starting and ending points, from which we can derive frame-level annotations. In this chapter, the multiple tracks of a clip are merged into one channel.

As there are no music auto-tagging dataset that contains frame-level annotations, I found that a multi-track instrument dataset like MedleyDB is especially suitable for evaluating the performance of music event localization. Common music auto-tagging datasets also contain a rich number of instrument tags. In our case, MagnaTagATune and MedleyDB have 46 tags in common. In addition, the multi-track audio of instruments in MedleyDB is also in line with the multi-label nature of music auto-tagging. Therefore, I propose to use MedleyDB for evaluating frame-level result.

doi:10.6342/NTU201801365

To ensure that the data are reasonably rich for both the training phase with MagnaTagATune and the evaluation phase with MedleyDB, the top 9 common instrument tags are chosen in our frame-level evaluation. These 9 tags are “drum set,” “electric bass,” “piano,” “male singer,” “clean electric guitar,” “synthesizer,” “vocalists,” “female singer,” and “acoustic guitar.”

3.2.1 Data processing

The audio feature used in our model is a 128-D log mel-spectrogram, which has been used in [12–14]. All features are standardized by removing mean and divided by standard deviation derived from training set. Features are extracted with a Python library Librosa [84] with 16k sampling rate and hop size 512.

In processing MagnaTagATune, we follow [13, 14] to use the first 12 sub-folders for training, the 13th sub-folder for validation and the 14th to 16th sub-folders for test. The training set is used to train the CNN models, the validation set to select the best parameters during training (see Section 3.3 for more information), and the test set to evaluate clip-level result. The union of training and validation sets is used to derive the frame-level thresholds.

In processing MedleyDB, we randomly divide the dataset into validation and test sets of roughly the same number of artists and the same number of clips. The artists in the validation set and the test set have no overlaps. There are 24 artists and 53 clips in validation set, and 25 artists and 55 clips in test set. The validation set is used to derive the frame-level thresholds and the test set is used to evaluate frame-level performance. Please note that MedleyDB is not used at all in training the proposed model, either for clip- or frame-level prediction.

doi:10.6342/NTU201801365

3.3 Experiments

The models are implemented with Lasagne[1] and Theano [85, 86]. The models are always trained with 188 output units, which is the total number of tags in MagnaTagATune. The predictions of top 50 tags and 9 instrument tags reported below are directly obtained from the 188-dimensional output. The FCNs are trained using back-propagation with binary cross-entropy as the loss function, 0.5 dropout rate, and 0.01 learning rate. Each early convolution layer consists of 32 filters, each of length 8. The late convolution includes three layers with 512, 512, and 188 filters, where 188 is the number of tags. The filter length and stride size of late convolutions are 1. The size of max pooling after a convolution layer equals the stride size of the convolution layer. We follow [13] to do convolution only along the temporal axis. The accumulation layer is also implemented with convolution whose initial weights form a centered Gaussian distribution. In order to keep the models with different total stride sizes to "see" temporal information of the same duration, the σs of Gaussian are initialized such that $TotalStride \times \sigma = 256$ frames. A model is trained with 200 epochs. The parameters from the epoch yielding the best performance on validation set is used as the final parameters for the model.

Because we train the models on MagnaTagATune and evaluate it on MedleyDB, we compare the performance using thresholds derived from MagnaTagATune or from MedleyDB in the way described in Section 3.1.2. The [0, 1] range is divided into $K = 1,000$ threshold candidates. Note that, in order to keep the assumption that we have only clip-level information, we divide the MedleyDB clips into smaller clips of 320 frames and aggregate the frame-level annotations into clip-level annotations and use them to derive the thresholds.

For brevity, some details of the model implementation are omitted. Please see the documentation of our codes for the details.

We will first see the best performance of clip-level and frame-level prediction in Section 3.3.2. The discussions in the subsequent subsections will use variants of the model with the best frame-level performance by changing the parameters or the type of a layer.

[1]https://lasagne.readthedocs.org/en/latest/

doi:10.6342/NTU201801365

Table 3.1: Best performance for the clip level and frame level. The baseline in frame-level performance is derived by simply predicting all frames as positive (i.e. all ones).

(a) Frame-level performance, comparing two ways of up-sampling.

	Precision		Recall		F1	
Up-sampling	Micro	Macro	Micro	Macro	Micro	Macro
Upscaling	0.561	0.539	0.846	0.791	0.675	0.633
Patching	0.560	0.538	0.847	0.793	0.674	0.633
Baseline	0.350	0.350	1.000	1.000	0.519	0.495

(b) Clip-level performance for the top 50 tags and all tags.

	Average AUC		MAP	
Top N tags	Per-class	Per-clip	Per-class	Per-clip
50	0.896	0.932	0.406	0.680
188 (All)	0.880	0.952	0.183	0.592

These variants help us gain insights into the properties of the frame-level model.

3.3.1 Metrics for Objective Evaluation

For evaluating clip-level result, average area under ROC curve (AUC) and mean average precision (MAP) are used. ROC curve is formed by connecting the points (true negative, true positive) from varied thresholds. Two type of variants are provided: 1) per-class, computing AUC or AP in a tag class first and then taking average over clips, and 2) per-clip, computing AUC or AP in a clip and then taking average over all tags.

For evaluating frame-level result, we use F1 score. In this study, the frame-level predictions of one tag in all test data are concatenated into one vector first for computing the measure scores. We use two ways to aggregate the measures from different tags. Micro F1 computes a precision score and a recall score globally from all tags and then computes F1 score, while macro F1 computes F1 score for each tag first and then takes average over all tags.

3.3.2 Best Performance

The best performance for the frame level is achieved by a model with an adaptive Gaussian filter, two early convolution layers of stride size 4 and a final mean pooling.

doi:10.6342/NTU201801365

Table 3.2: Effect of thresholds for frame-level predictions. Thresholds are derived either from MedleyDB or MagnaTagATune. "Early conv." refers to the number of layers in early convolutions.

Final pooling	Early conv.	Threshold source	Precision Micro	Precision Macro	Rrecall Micro	Rrecall Macro	F1 Micro	F1 Macro
Max	2	MedleyDB	0.530	0.523	0.830	0.806	**0.647**	**0.617**
		MagnaTagATune	0.554	0.506	0.150	0.215	0.236	0.241
	3	MedleyDB	0.462	0.447	0.916	0.883	**0.614**	**0.578**
		MagnaTagATune	0.546	0.566	0.182	0.245	0.273	0.261
Mean	2	MedleyDB	0.561	0.539	0.846	0.791	**0.675**	**0.633**
		MagnaTagATune	0.606	0.489	0.221	0.273	0.324	0.300
	3	MedleyDB	0.498	0.474	0.888	0.844	**0.638**	**0.594**
		MagnaTagATune	0.530	0.540	0.183	0.244	0.273	0.258

The thresholds are derived from MedleyDB. As Table 3.1a shows, using upscaling and using patching for up-sampling have little difference. As patching is computationally more demanding, only the results using upscaling will be reported in the following experiments.

The best performance for the clip level with top 50 tags, on the other hand, is achieved by a model which has a fixed Gaussian filter, one early convolution layer of stride size 4 and a final max pooling. The average AUC achieved by this model is 0.896, which is comparable to the multi-scale model proposed by [14], which achieving 0.898 average AUC. The result using all 188 tags is also listed in Table 3.1b. The clip-level observations with 50 top tags also generally hold with 188 top tags. For space limit, only the results with 50 top tags will be reported regarding clip-level hereafter.

3.3.3 Effect of Thresholds

As the first investigation, we compare the thresholds for frame-level predictions derived from either MagnaTagATune or MedleyDB. The results are shown in Table 3.2. The thresholds from MedleyDB yield better recall scores and accordingly F1 scores than those from MagnaTagATune. This indicates that we still need some data in the target domain for deriving thresholds in order for this transfer learning scheme to work. In the following experiments, the reported frame-level predictions are derived with MedleyDB thresholds.

doi:10.6342/NTU201801365

3.3.4 Effect of Accumulation

In this set of experiments, we compare the effect of accumulation layer. It is either no accumulation, fixed Gaussian filter, or adaptive Gaussian filter. The performance is shown in Table 3.3. Most of the models with accumulation outperform the others without accumulation in the same category significantly (one-tailed t-test on macro F1 with 0.05 significance level). We first notice that the clip-level performance without accumulation is comparable or even better than the performance with Gaussian filters. However, the frame-level performance without Gaussian filters is worse than those with Gaussian filters. The performance is even worse than the baseline in Table 3.1a in some case.

None of adaptive Gaussian and fixed Gaussian has clear advantage over the other. As our focus in this study is frame-level prediction and the adaptive Gaussian yields the best frame-level performance, we will focus on the performance of the adaptive Gaussian in the remaining discussion.

To visually see the effect of accumulation, Figure 3.3 shows the output of four models: adaptive Gaussian filter with total stride 256, adaptive Gaussian filter with total stride 2, no accumulation with total stride 256, and no accumulation with total stride 2. When no accumulation is applied, the model with small total stride has rapid changes in amplitude and loses continuity of the prediction. In contrast, when Gaussian filters are used, the predictions are smooth with either big or small total stride.

3.3.5 Effect of Multi-scale Input

We compare the performance of input of one scale (window size=1024), inputs of 3 scales (window sizes={1024, 4096, 16384}), and inputs of 6 scales (window sizes={512, 1024, 2048, 4096, 8192, 16384}). We can see in Table 3.4 that the multi-scale inputs yield better results than inputs with a single scale. Inputs with 6 scales yield the best results at either clip-level or frame-level, so we will use 6 scales in all the remaining experiments. Only the results with adaptive Gaussian are shown in Table 3.4, but the pattern also holds with the fixed Gaussian.

doi:10.6342/NTU201801365

Table 3.3: Accumulation layers. The models either use no accumulation, use accumulation with Gaussian of fixed standard deviations (σs), or use accumulation with Gaussian of adaptive standard deviations (σs). 50 classes for clip-level and 9 classes for frame-level

			Frame level		Clip level	
Final	Early	Accum.	F1		Average AUC	
pooling	conv.	(Gaussian)	Micro	Macro	Per-class	Per-clip
Max	2	None	0.401	0.313	0.878	0.922
		Fixed	**0.654**	0.606	**0.891**	0.928
		Adaptive	0.647	**0.617**	**0.891**	**0.931**
	3	None	0.601	0.525	0.885	0.925
		Fixed	**0.648**	**0.621**	0.886	0.926
		Adaptive	0.614	0.578	**0.894**	**0.930**
Mean	2	None	0.557	0.509	0.893	0.931
		Fixed	0.595	0.558	**0.894**	**0.932**
		Adaptive	**0.675**	**0.633**	0.893	0.931
	3	None	0.591	0.533	**0.896**	0.930
		Fixed	**0.651**	**0.606**	0.894	**0.931**
		Adaptive	0.638	0.594	0.891	0.929

Table 3.4: Effect of multi-scale inputs. Inputs of 1 scale, 3 scales, or 6 scales are experimented.

		F1		Average AUC	
Early conv.	# of inputs	Micro	Macro	Per-class	Per-clip
2	1	0.605	0.585	0.882	0.921
	3	0.628	0.608	0.892	0.930
	6	**0.675**	**0.633**	**0.893**	**0.931**
3	1	0.628	0.585	0.884	0.922
	3	**0.642**	**0.606**	**0.892**	0.928
	6	0.638	0.594	0.891	**0.929**

3.3.6 Resolution and Performance Trade-off

The issue of trade-off between clip-level performance and frame-level performance with CNNs is known in other fields. It has been pointed out [5] that “the pooling and sampling may affect label localization through time because it decreases time resolution for higher CNN layers” in the context of speech recognition. It also happens in visual object recognition [87]: “There is a natural trade-off between classification accuracy and localization accuracy with convolutional networks: Deeper models with multiple max-pooling layers have proven most successful in classification tasks, however their increased invariance and large receptive fields make the problem of inferring position from the scores at their top output levels more.”

doi:10.6342/NTU201801365

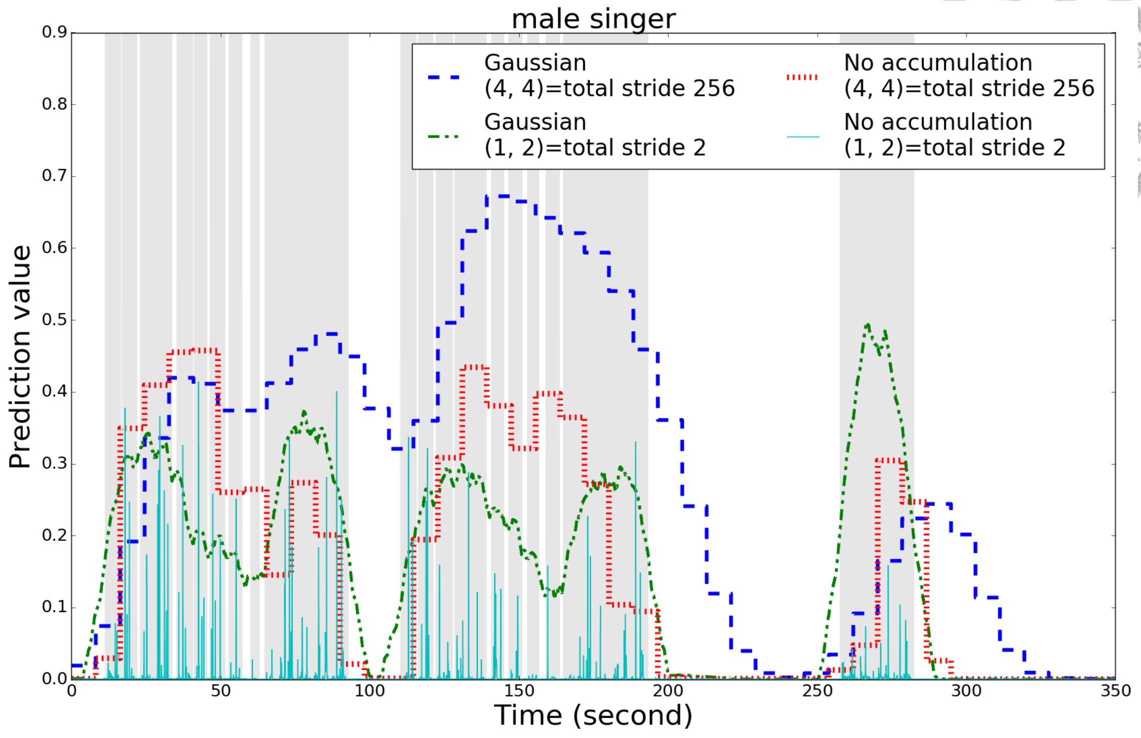


Figure 3.3: Smoothing effect of accumulation layers for a music piece from MedlyDB. The frame-level predictions of four models with final max-pooling are shown. The micro F1 scores achieved by the four settings (Gaussian, stride 2), (Gaussian, stride 256), (No accumulation, stride 2) and (No accumulation, stride 256) are 0.658, 0.601, 0.158, and 0.601, respectively. This is the music piece entitled AlexanderRoss_GoodbyeBolero in the MedlyDB dataset.

We can also see this effect in music auto-tagging from our experiments. By controlling the number of early convolution layers, we also control the total stride size, which determines the resolution of the output. We can see a pattern in Table 3.5. When we use Gaussian for the accumulation layer, the clip-level performance improves as the total stride size increase while the frame-level performance gets worse. On the other hand, when we do not use accumulation, the performance of both clip-level and frame-level gets worse as the total stride increases. The pattern of the performance without Gaussian follows our observation in the Section 3.3.4 that frame-level predictions would be more difficult without accumulation.

3.3.7 Effect of Final Pooling Functions

Two pooling functions are tested for the final layer, mean-pooling and max-pooling. From Table 3.5, there is no obvious advantage over the other using max and mean pool-

doi:10.6342/NTU201801365

Table 3.5: Effect of different total stride sizes. The total stride is related the resolution of the CNN. The larger the total stride, the worse the output resolution. We control the total stride by controlling the number of early convolutional layers. The stride size is 4 for every early convolution layer.

	Final	# of early		F1		Average AUC	
Accumulation	pooling	conv layers	total stride	Micro	Macro	Per-class	Per-clip
None	Max	1	4	0.189	0.141	0.872	0.915
		2	16	0.401	0.313	0.878	0.922
		3	64	**0.601**	0.525	**0.885**	**0.925**
		4	256	**0.601**	**0.554**	0.885	0.923
	Mean	1	4	0.488	0.463	0.886	0.926
		2	16	0.557	0.509	0.893	**0.931**
		3	64	0.591	0.533	**0.896**	0.930
		4	256	**0.614**	**0.562**	0.889	0.926
Adaptive Gaussian	Max	1	4	**0.658**	**0.637**	0.887	0.926
		2	16	0.647	0.617	0.891	**0.931**
		3	64	0.614	0.578	**0.894**	0.930
		4	256	0.601	0.541	0.889	0.926
	Mean	1	4	0.650	0.628	0.885	0.925
		2	16	**0.675**	**0.633**	**0.893**	**0.931**
		3	64	0.638	0.594	0.891	0.929
		4	256	0.600	0.540	0.891	0.926

ing. However, in principle, max pooling should be used if the training data have different durations to have comparable outputs.

3.3.8 Learned Parameters of Gaussian Filters

Figure 3.4 shows the standard deviation σs of the Gaussian filters for the top 50 popular classes. The parameters shown here are taken from the model with the best frame-level performance, the one presented in Section 3.3.2. The x-axis is the number of training samples in the training data, and the y-axis is the learned σs. The first thing we can notice visually is a downward trend as the number of training data increases. Indeed, the Pearson's correlation coefficient between σs and the number of training data is -0.442, so they are moderately linearly correlated. The linear regression line is also shown in Figure 3.4 with green dashed line.

Although linearly correlated to σ in some degree, the number of training data is not the only factor affecting σs. The labels in red color are instruments, and the labels in

doi:10.6342/NTU201801365

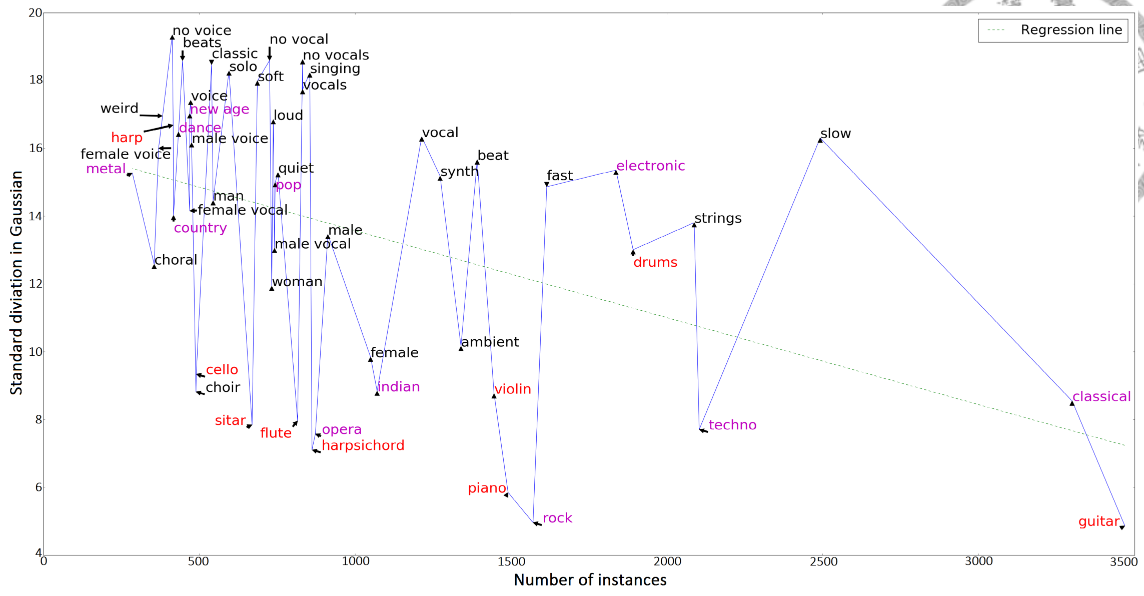


Figure 3.4: The standard deviation σ in Gaussian filter. x-axis is the number of instances in the training data, and y-axis is the quantity of σ of the classes after training. Instrument tags are in red color, and genre tags are in purple. The green dashed line is the linear regression line of the points on the image.

purple color are genres. We can see that the σs of instruments are in general smaller than the σs of genres in Figure 3.4. This verifies our intuition that the instruments require smaller durations of time to recognize while genres are more complex and require larger durations to recognize. We can also see that the tempo-related tags, “fast” and “slow,” and the negation tags, “no vocal” and “no vocals,” have larger σs, which makes sense because recognition of tempo and determining the non-existence of properties requires longer duration.

Another thing one may wonder is whether these learned parameters could be somewhat random. We observed that the learned parameters from models with varied total stride sizes show similar patterns. To see this quantitatively, we compute the Pearson’s correlation coefficient between σs of the top 50 tags from different models. First, the correlation coefficients between σs from the best model and σs from its counterparts of (stride size, # of early conv. layers)=(4, 1), (4, 3), and (4, 4) are 0.990, 0.957, and 0.924, respectively. Second, the correlation coefficient between σs from the best model and σs from its counterpart using max as final pooling is 0.958. The σs are highly correlated between different models. This result together with the observation from last paragraph

doi:10.6342/NTU201801365

Table 3.6: The performance of models trained directly with frame-level annotations. The number under "late layers" represents the number of units used in a layer. If two numbers are present, there are two layers.

Frame model	Precision		Recall		F1	
late layers	Micro	Macro	Micro	Macro	Micro	Macro
128	0.572	0.481	0.627	0.549	0.598	0.471
512	0.565	0.485	0.632	0.548	0.597	0.469
128, 128	0.571	0.480	0.617	0.540	0.593	0.468
512, 512	0.560	0.478	0.634	0.555	0.595	0.471

suggests that the patterns are not random.

3.3.9 Comparing with Frame-to-Frame training

We compare the proposed model with a model trained directly with frame-level annotations from MedleyDB. Neural networks (FCN with pooling_window equals 1) are built to conduct frame-to-frame training. We further separate 80% of clips from the validation set in MedleyDB as the training set. The results are shown in Table 3.6. Our proposed model significantly outperforms these frame-to-frame models (one-tailed t-test on macro F1 with 0.05 significance level).

There are several possible reasons for this counter-intuitive result. First, frame-to-frame training is itself a complex research problem, the models tested here might not be optimal and might not be the best structure. Second, despite of the large number of frames for training, the number of unique clips in the training set is small. In the training set, the number of clips in each class is from 5 to 24 (13 clips per class in average). Furthermore, the MedleyDB data are divided in a more strict way such that the artists in the training set and the artists in the test set do not overlap. Therefore, the networks do not see enough data for generalization. In contrast, in MagnaTagATune, the number of clips in each class is from 37 to 1,892 (779 clips per class in average). This conjecture is also supported by the observation that there is a big gap between the validation loss and training loss (training loss is much smaller than validation loss) during the training phase even though we have used dropout and used smaller hidden layers.

This indicates that the proposed clip-to-frame training scheme could be a more realistic

doi:10.6342/NTU201801365

solution to this problem.

3.4 Visualization of the frame-level predictions

In this section, the frame-level predictions from the best frame-level model are visualized to subjectively evaluate the frame-level prediction. Examples from the test data of the two datasets are shown. The pre-accumulation layer in the figures refers to the layer right before the accumulation layer. In order to investigate the prediction result subjectively, I have also made a demo website (http://clip2frame.ciaua.com). A discussion about the website and the observations I have made from using the demo website is presented in Appendix A.

3.4.1 MedleyDB

Examples are shown in Figure 3.5 and Figure 3.6. The examples in Figure 3.5a and 3.5b have good frame-level predictions and the threshold cuts at roughly the right place. The pre-accumulation output in Figure 3.6a looks reasonable, but the threshold cuts at a wrong place. Figure 3.6b has totally wrong frame-level prediction visually. By listening to the audio, we realize that the activated parts predicted by our model are in fact also electric guitar but are distorted electric guitar, not a clean electric guitar. In MagnaTagATune, there is an "electric guitar" tag, but the distorted and the clean electric guitar are not distinguished. We have used the output dimension of "electric guitar" in MagnaTagATune for evaluating frame-level "clean electric guitar" in MedleyDB. In MagnaTagATune, there might be less clean electric guitar and more distorted electric guitar, so the model cannot recognize the sound of a clean electric guitar.

3.4.2 MagnaTagATune

Figure 3.7 shows one example from MagnaTagATune test data. Figure 3.7a is the predictions derived from the accumulation layer, and Figure 3.7b is the predictions from the pre-accumulation layer. By listening to the original clip, we know that there is a man

doi:10.6342/NTU201801365

talking from 0 to 4 seconds, and then drums appear. Electrical guitar starts from 8 seconds and the drums continue. At this point, it starts to sound like a rock song. That is why we see the pattern in the figure. Even though “rock” is a genre tag which is usually annotated on the whole clip, we can also consider it as a music event that only exists in some parts of the music.

3.5 Application to audio event detection

Audio event detection (AED) shares a similar problem as music event detection, that is, the lack of detailed annotations for training models. Therefore, we also extend the proposed methodology to weakly-supervised audio event detection [75], where I am the second author. In that paper, the model is basically the same one used in music event detection proposed by me and the experiments are done by Ting-Wei Su. In the remaining of this section, the main results are presented. For details of this work, please refer to the original paper [75].

The goal of audio event detection (AED), or sound event detection, is to detect sound events in daily lives, such as screaming, shouting, and gun-shots for security [88–90] as well as breath and snore for medical purpose [91].

We build and evaluate such models with UrbanSound and UrbanSound8K datasets [92].

The model is different from the one used for music event detection in two aspects. First, max pooling is used for global pooling because audio events are often pretty short compared with music events. Using average pooling could include too many false positives and degrades the performance. Second, data augmentation is used because audio events in the real world have varied intensities.

Similar to music events, audio events may have different temporal properties. Some events happen in merely a few frames, like dog bark, while others can last for several minutes, such as the sound generated by an air conditioner in operation. To deal with it, a Gaussian filter layer is inserted between the final later convolutional layer and the final max-pooling, as what has been done for music audios.

doi:10.6342/NTU201801365

This Gaussian filter layer is implemented with convolution, but the filter weights are set to fit a centered Gaussian distribution:

$$g[t] = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{t^2}{2\sigma^2}}, \tag{3.1}$$

where σ is the standard deviation in the Gaussian distribution. g is applied to the signal s and scan through the signal:

$$(s \cdot g)[n] = \sum_{m=\frac{-M}{2}}^{\frac{M}{2}} s[m] \cdot g[n-m], \tag{3.2}$$

where M is a pre-determined filter size.

3.5.1 Datasets

Two datasets are employed in the experiments: UrbanSound and UrbanSound8K. UrbanSound is used for training and frame-level evaluation, and UrbanSound8K is used for clip-level evaluation. Both datasets are composed of ten different classes of sounds came from Freesound.org. The ten classes and the number of data can be seen in Table 3.7. Every clip in these datasets contains only one label.

UrbanSound dataset comprises 1302 recordings with their durations varying from 1 second to over 30 seconds. All audio clips are annotated with the onset and offset of the sound events appearing in them. Because of these annotations, we are able to evaluate frame-level predictions. However, notice that these temporal annotations are used only for evaluating testing result. We do not use them in the training phase. Although the size of UrbanSound is not large enough, it is still the most suitable public dataset for our work.

UrbanSound8K is composed of 8732 short clips segmented from files in UrbanSound. Files in UrbanSound8K are less than or equal to 4 seconds and is labelled with one class. It is used for clip-level evaluation.

doi:10.6342/NTU201801365

Table 3.7: This table provides the number of data in each dataset, the learned standard deviations σ of Gaussian filter layer after training, and the performance of our model. The US dataset evaluated with AUC score represents the performance of frame-level predictions, and the accuracy tested on US8K is the clip-level evaluation result.

	# of data			AUC	Acc.
Class	US	US8k	σ	US	US8k
Air conditioner	64	1000	2.93±0.30	0.612	76.7%
Car horn	125	429	1.32±0.05	0.807	69.0%
Children playing	158	1000	1.08±0.07	0.594	41.8%
Dog bark	337	1000	0.96±0.06	0.790	79.5%
Drilling	119	1000	2.05±0.13	0.764	52.3%
Engine idling	97	1000	2.97±0.19	0.688	51.8%
Gun shot	117	347	1.13±0.16	0.921	94.4%
Jackhammer	45	1000	1.93±0.10	0.704	39.8%
Siren	74	929	2.09±0.17	0.763	59.0%
Street music	166	1000	1.51±0.10	0.737	61.3%
Total	1302	8732		0.738	59.4%

3.5.2 Experiments

There are two major parts in the experiment. The first part shows the clip-level prediction of the proposed model in different structures and compares the best result with a fully-supervised work which was also tested on UrbanSound8K. In the second part, we will evaluate the result of frame-level prediction.

The sampling rate of audio files is set to 44100 Hz. The input feature is composed of an 128-dimension mel-spectrogram and its first derivative, which is also 128 dimensions. Our model contains 2 early convolutional layers. Each layer comprises 60 filters with filter size 5 in time domain. We follow [13] and do convolution only on time domain. Therefore, only the first convolutional layer has its filter size 128 in frequency domain while the others have a filter size of 1. Each early convolutional layer is followed by a max-pooling layer with both pooling size and stride size being 4 in temporal axis (Pooling is not done on the frequency axis). The late layers include 3 consecutive convolutional layers with their filter size being 1. We set the filter number to 128 for first two layers and 10 for the last one, which is the total number of the labels in UrbanSound. As for the Gaussian filter layer, the filter size is set to 32, and the initial standard deviation σ of every class to 2. All dropout rates in this model are set to 0.5, and the learning rate is initialized

doi:10.6342/NTU201801365

Table 3.8: The comparison between the clip-level accuracy of basic setting and of adding one of multiple scales, data augmentation, and Gaussian filter to the model.

Basic	w/ Multiple scales	w/ Data augmentation	w/ Gaussian filter
25.93%	37.51%	39.78%	51.73%

to 0.006. Adaptive Gradient Algorithm (AdaGrad) is used as our update method [93], so the learning rate will be changed every time we update the parameters. As the training data consist of clips of varied duration, the batch size is set to 1 to simplify the training process. 300 epochs are run in every training set, and the model belonging to the epoch with highest validation accuracy will be selected as the final model for an experimental setting.

Clip-level Evaluation

We begin with evaluating the following three modifications of the CNN model. First, in the basic structure, the window size of STFT is set to 1024. In the multi-scale setting, we instead use a structure of 3-scale input feature with window sizes being 1024, 4096, and 16384. As we see in the Table 3.8, multi-scale model outperforms the basic one with a large margin. Second, owing to the fact that weakly-supervised data may vary quite a lot in the volume of audio events, and that the training data are scarce from UrbanSound, we augment the training data by adding and reducing 5db to every clip. Thus, the training data are tripled and should make the model less sensitive to the effect of diverse volumes. The result shows that it does improve the performance after data augmentation on volume. Third, the Gaussian filter described is added.

We refer to the model with only single-scale input, no augmentation and no Gaussian filter as the 'basic' model. We than add one of these three modification to the model and investigate which one can more effectively improve the basic model. The result is shown in Table 3.8. We can see that the multi-scale feature is indeed beneficial, improving the basic model by a margin. In addition, the use of data augmentation is also quite effective. More importantly, we found that the Gaussian filter largely improves the performance. Furthermore, the final standard deviations of the Gaussian filters provide insights regard-

doi:10.6342/NTU201801365

ing the classes. From Table 3.7, we can see that sound classes with longer durations, such as "air conditioner" and "engine idling," obtain higher values, while classes with shorter durations, like "gun shot" and "dog bark," get lower values. Therefore, if "gun shot" and "engine idling" are both detected in a very short duration, gun shot is more likely to be highlighted by the Gaussian filter layer. On the contrary, the final prediction of engine idling will be reduced since this kind of sound is supposed to occur in a longer duration.

Finally, we turn on all three functions. As shown in Table 3.7, our model attains 59.4% accuracy, which is better than the result of using Guassian filter alone. In a fully-supervised setting, Piczak achieved 73.1% accuracy by training on subsets of the UrbanSound8K itself. Although there is still a performance gap, this result is promising for it only uses weakly-supervised data from UrbanSound.

Frame-level Evaluation

The frame-level result is evaluated with average area under ROC curve (AUC) [94], and Table 3.7 shows the overall result and the result of each class. In addition, some visualized frame-level results are shown in Fig. 3.8. The ability of localizing events can be well seen. In general, when an event is detected, their temporal locations are usually correct.

3.6 Summary

In this paper, a model has been proposed that can make frame-level predictions with only clip-level annotations in the training phase. The result demonstrates that this weakly-supervised learning approach for music event localization is possible, and it achieves 0.675 micro F1 score evaluated on MedleyDB. We have found that the accumulation layer implemented with Gaussian filter can help improve the frame-level prediction and it also captures the characteristics of the music events.

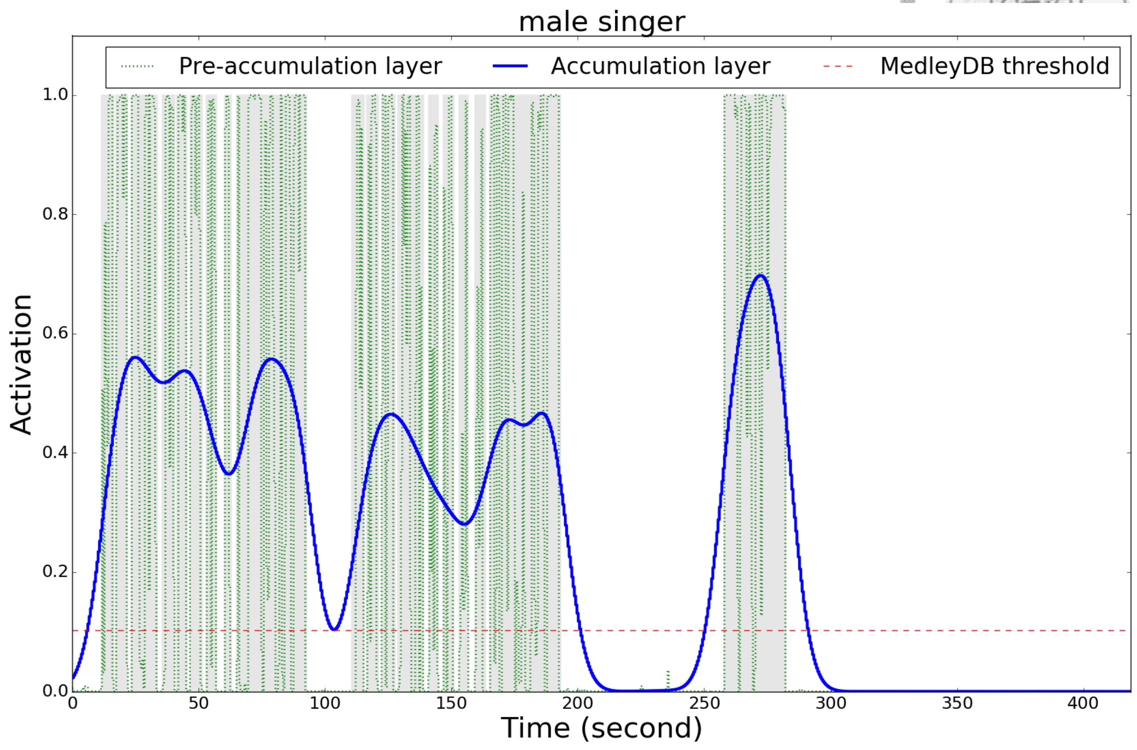
The initial σs may require more experimentation in the future. Observing the downward trend with the number of training data in Figure 3.4, it is possible that the initial σs

doi:10.6342/NTU201801365

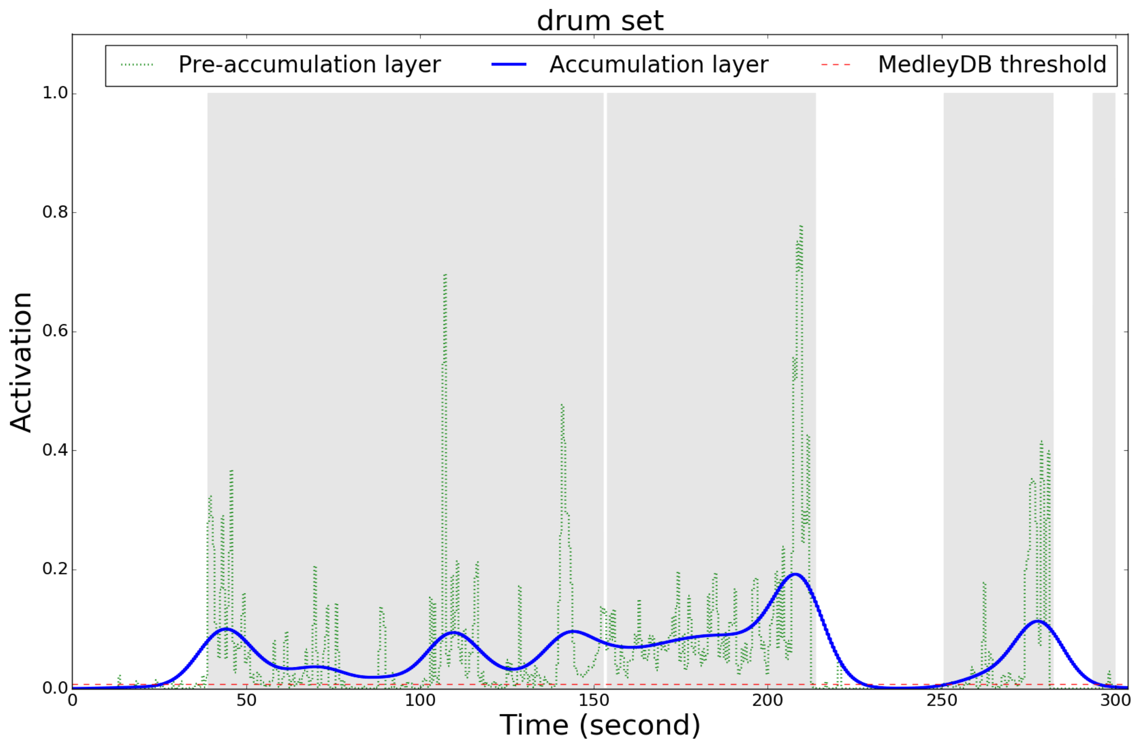
are too large. If the initial σs are too small or too large, it is difficult for classes with fewer data to learn good σs.

The number and diversity of the training dataset is an important issue. We have argued in Section 3.3.9 that the frame-to-frame training might not yield better result than our approach because the training data used by the frame-to-frame training is not diverse enough. In addition, we have also observed in Section 3.4.1 that the training data are also not diverse enough to include different types of electric guitar sounds even with our approach. This indicates that it is necessary to collect more diverse data in order to further improve the frame-level performance.

doi:10.6342/NTU201801365



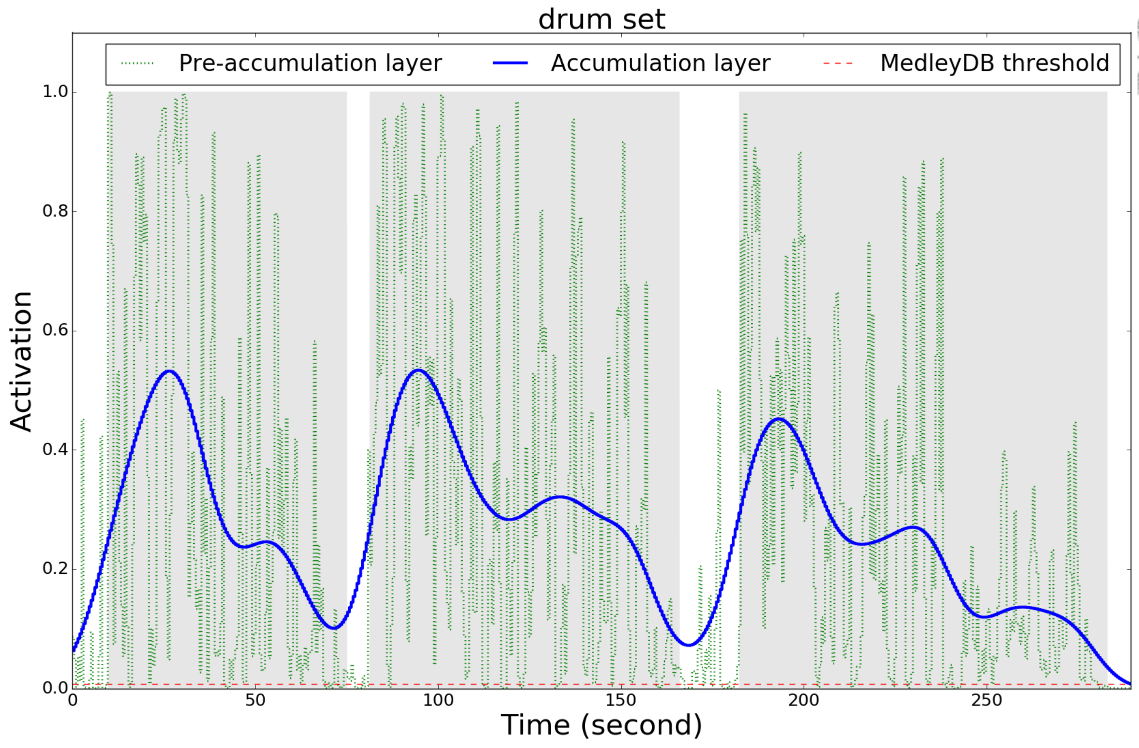
(a) Good. "male singer" from AlexanderRoss_GoodbyeBolero



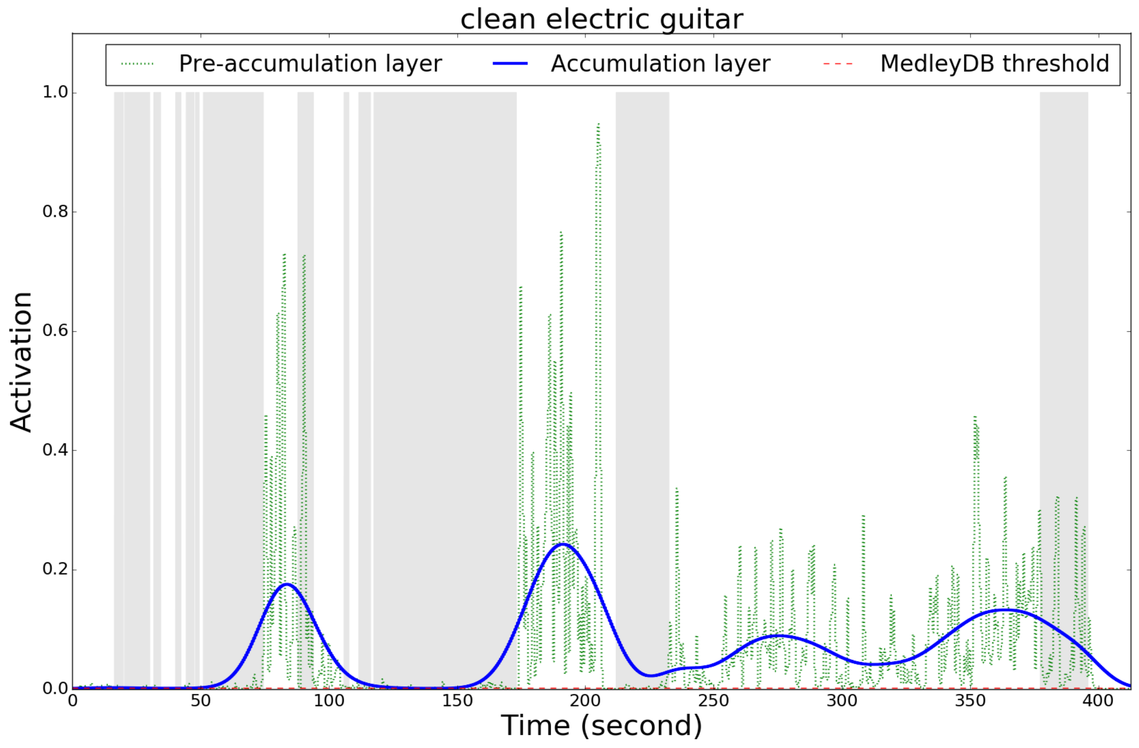
(b) Good. "drum set" from TheScarletBrand_LesFleursDuMal

Figure 3.5: Visualization of frame-level predictions on MedleyDB test data: Good examples.

doi:10.6342/NTU201801365



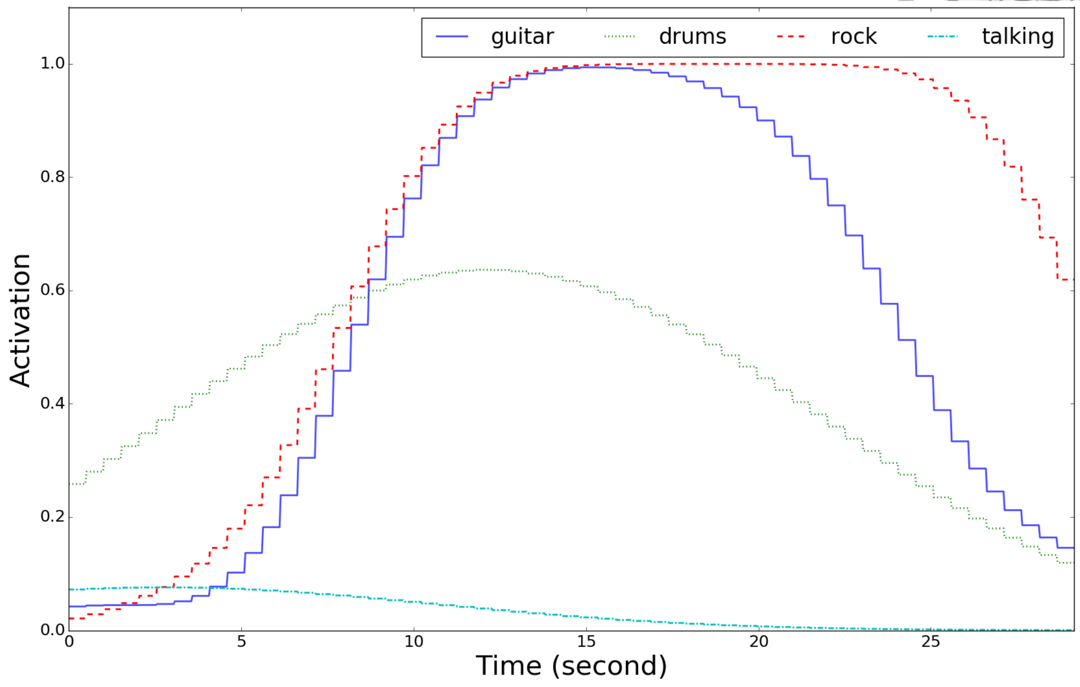
(a) Bad. “drum set” from StevenClark_Bounty



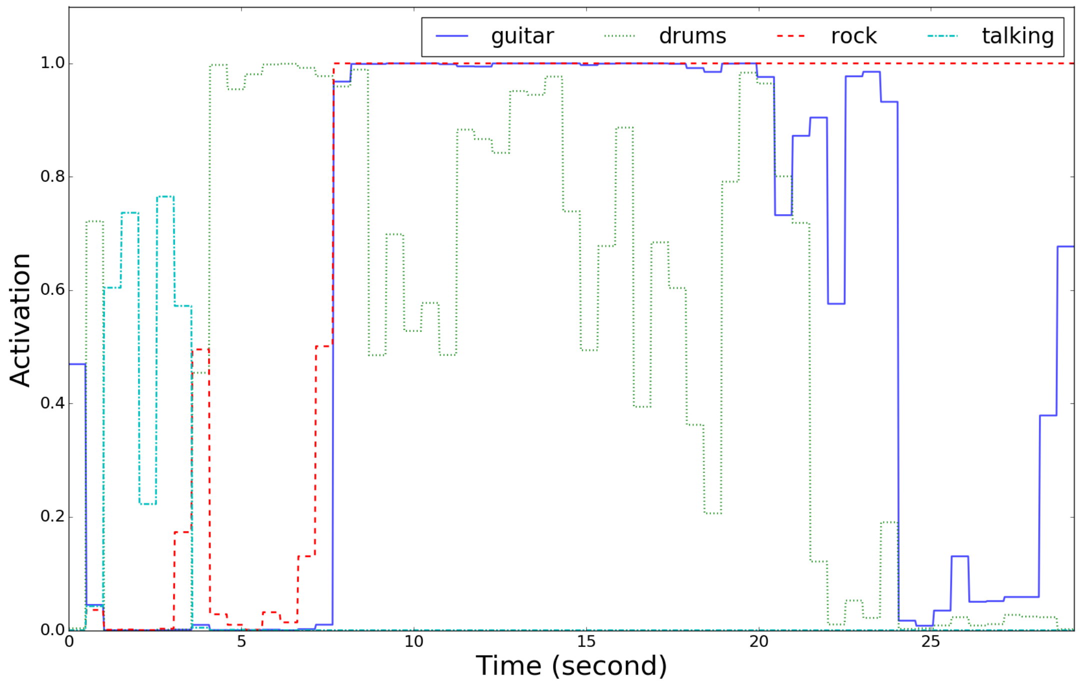
(b) Bad. “clean electric guitar” from Meaxic_YouListen

Figure 3.6: Visualization of frame-level predictions on MedleyDB test data: Bad examples

doi:10.6342/NTU201801365



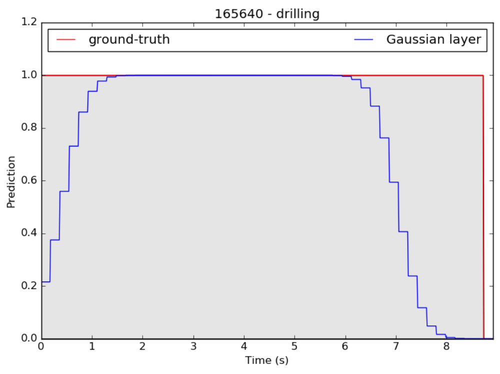
(a) Accumulation



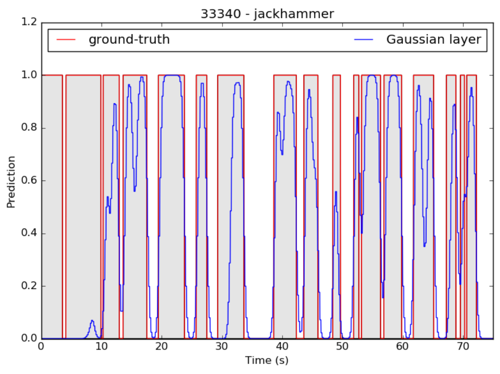
(b) Pre-accumulation

Figure 3.7: Visualization of frame-level predictions on MagnaTagATune test data. They show predictions of four tags annotated on the clip. The outputs from accumulation layer and from pre-accumulation layer are shown. Both of them are from the same model. Title: jackalopes-jacksploitation-04-kentucky_applejack-0-29.

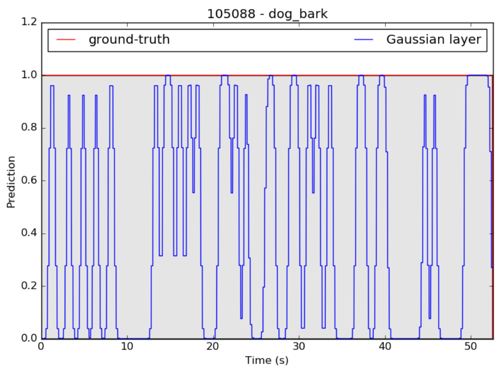
doi:10.6342/NTU201801365



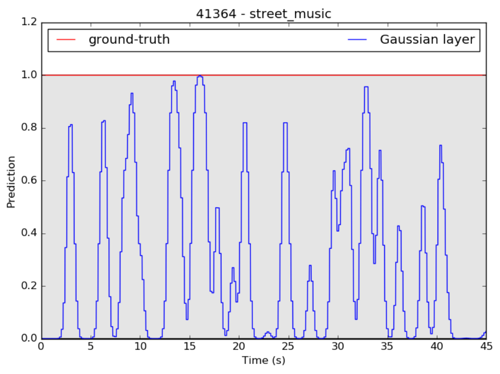
(a) Drilling of 165640.wav



(b) Jackhammer of 33340.wav



(c) Dog bark of 105088.wav



(d) Street music of 41364.wav

Figure 3.8: Visualized frame-level results. Red line is the ground-truth, and blue denotes frame-level prediction.

doi:10.6342/NTU201801365

Chapter 4

Weakly-supervised Visual Instrument-playing Action Detection in Videos

In this chapter, we will see how one can detect instrument-playing action in videos with weak labels. The proposed method is based on two observations:

Action-in-object The spatial location of an action is close to (e.g. at the border or within) the spatial location of objects (e.g., instruments, bats, balls, or weapons).

Action-making-sound A specific type of actions is associated with a specific type of sounds that the actions make.

We will use two auxiliary models to provide supervisory signals to the action model. The sound model provides temporal supervision, and the object model provides spatial supervision. These two auxiliary models are themselves trained with weak labels.

This chapter is organized as follows. In Section 4.1.2, a training framework is proposed to learn the temporal and spatial locations of the actions by utilizing an auxiliary object model and an auxiliary sound model. In Section 4.1.3, how to fuse the object and sound information is introduced to further improve the result after the action model is trained. In Section 4.2.3, the datasets used in this chapter is introduced. Among them is

doi:10.6342/NTU201801365

a dataset annotated by myself with totally 5,400 frames from 135 videos with detailed locations of instrument-playing actions. Extensive experiments and analyses are conducted in Section 4.3. Finally, this chapter is summarized in Section 4.4.

4.1 Proposed method

In what follows, we will firstly see the proposed weakly-supervised method by increasing supervisions in the process of training the action model by information from other modalities. Then, we will see how we can fuse information from other modalities, after the action model has been trained.

4.1.1 Instrument-playing actions

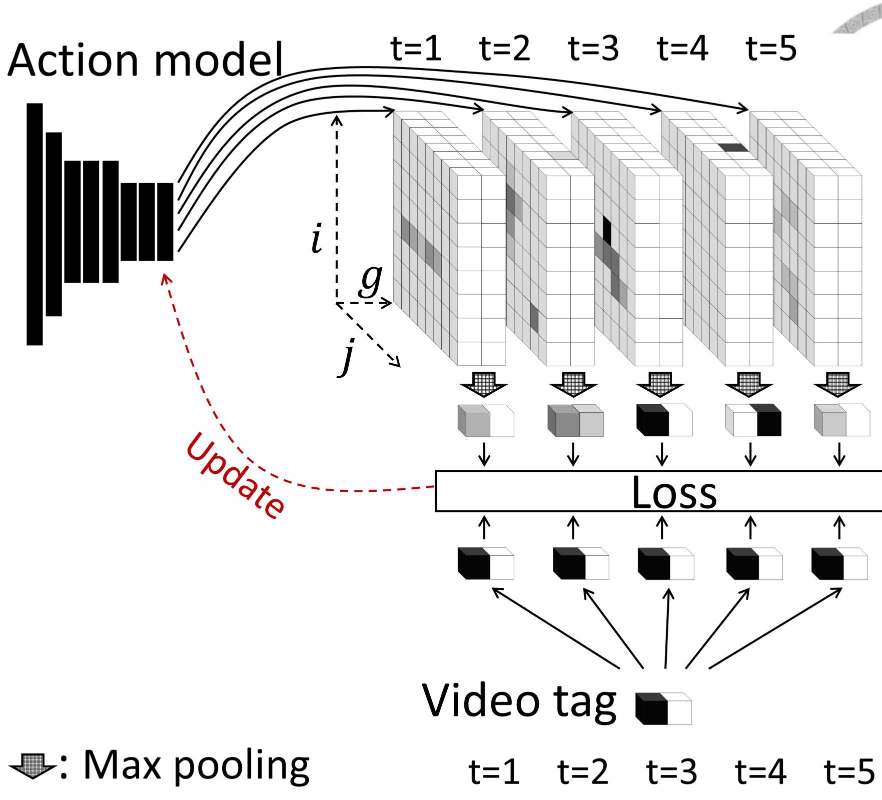
There could be various movements in the body or the instrument when a musician plays an instrument. For example, when a pianist is playing piano, the hands, arms, and the whole body could be all moving. However, we do not consider all of them as instrument-playing actions.

In this chapter, "instrument-playing actions" or "playing actions" refers to the movements or actions that are most directly responsible for making the instrument sounds in the scene. For example, the playing actions of flute include the finger pressing movements over the flutes and the blows with mouths, while the playing actions of guitar include the pressing of the fretting hand and the string picking/strumming of the other hand. Although other movements such as arm movements and body movements are also important in instrument playing and music performance, they will not be considered as instrument-playing actions in this chapter with the above meaning of "instrument-playing actions".

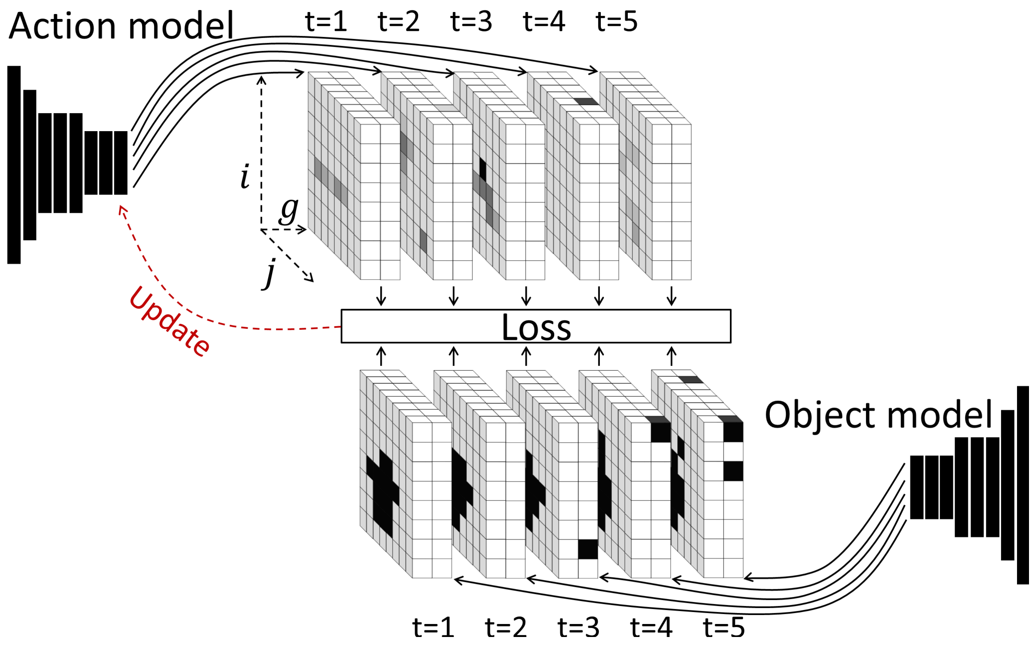
4.1.2 Increasing supervisions for training the action model

The goal of this chapter is to determine whether playing actions are present, which instruments correspond to the playing actions, and where the playing actions are in the scene. An intuitive way to build an instrument-playing action detector is to utilize the

doi:10.6342/NTU201801365



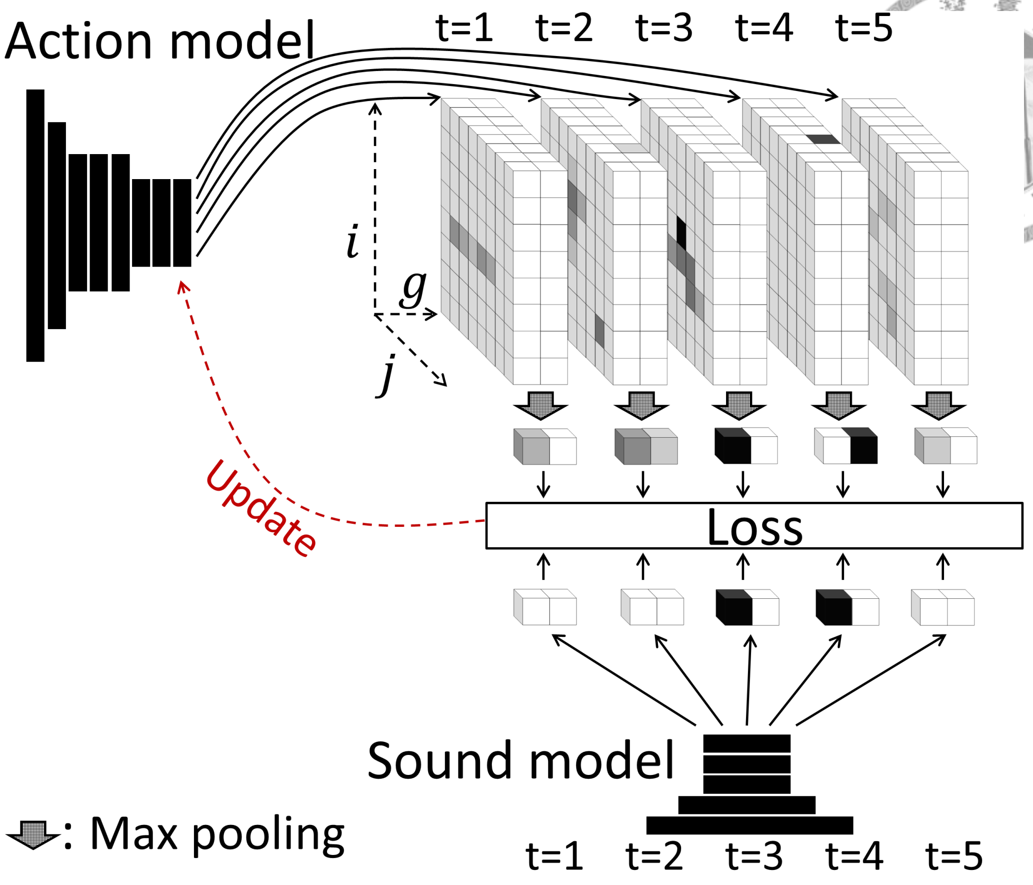
(a) Video tag as target (VT): $loss(\max_{i,j} A_{g,t,i,j}, V_g)$



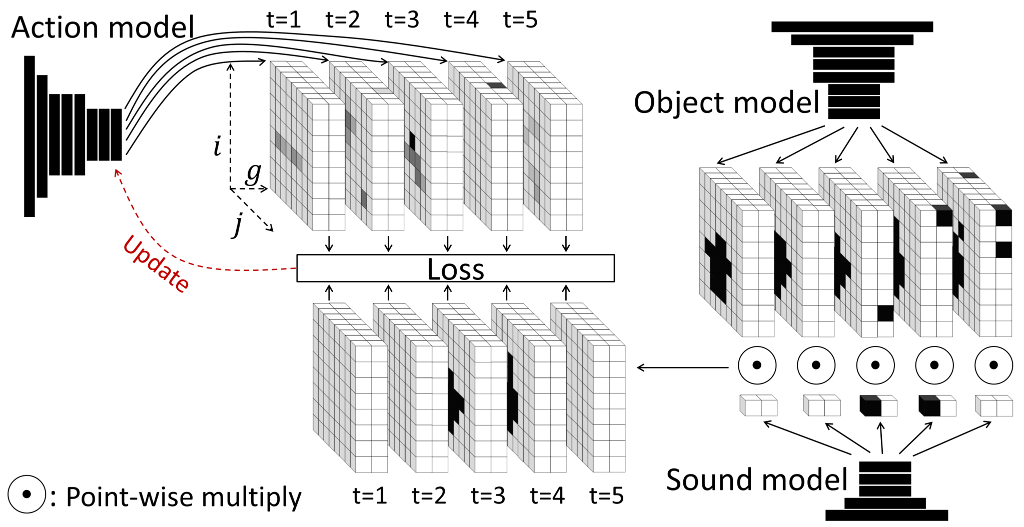
(b) Object as target (OT): $loss(A_{g,t,i,j}, \hat{O}^u_{g,t,i,j})$

Figure 4.1: Four levels of supervisions: Video tag as target (VT) and Object as target (OT) (continued in Figure 4.2). g is the instrument index, t is the temporal index, and (i, j) is the spatial coordinate in the output map. $[V_g]$ is the vector of the video tags, $[\hat{O}^u_{g,t,i,j}]$ is the binarized output of the object model, $[\hat{S}^v_{g,t}]$ is the binarized output of the sound model, and $[A_{g,t,i,j}]$ is the output of the action model. Darker shade represents higher activation where activation values are between 0 and 1. Each cuboid in the figure represents an output tensor produced by a model at a given time t, and the three axes of a cuboid represent the instrument index g and the spatial coordinate (i, j), respectively. The t above or below a cuboid represents its temporal index.

doi:10.6342/NTU201801365



(a) Sound as target (ST): $loss(\max_{i,j} A_{g,t,i,j}, \hat{S}^v_{g,t})$



(b) Sound×Object as target (SOT): $loss(A_{g,t,i,j}, \hat{S}^v_{g,t}\hat{O}^u_{g,t,i,j})$

Figure 4.2: Four levels of supervisions: Sound as target (ST) and Sound×Object as target (SOT). g is the instrument index, t is the temporal index, and (i, j) is the spatial coordinate in the output map. $[V_g]$ is the vector of the video tags, $[\hat{O}^u_{g,t,i,j}]$ is the binarized output of the object model, $[\hat{S}^v_{g,t}]$ is the binarized output of the sound model, and $[A_{g,t,i,j}]$ is the output of the action model. Darker shade represents higher activation where activation values are between 0 and 1. Each cuboid in the figure represents an output tensor produced by a model at a given time t, and the three axes of a cuboid represent the instrument index g and the spatial coordinate (i, j), respectively. The t above or below a cuboid represents its temporal index.

doi:10.6342/NTU201801365

instrument tags accompanying the videos for model training. The tags can be either from a video dataset or from the titles of the videos on an online-streaming website. We may assume that each frame contains the playing actions of this tag g somewhere in the scene. Therefore, while training the action model, we can apply a spatial max-pooling to the output of the action model and compare it with the tag of the video clip, $V = [V_g]$, by a loss function, that is, $loss(\max_{i,j} A_{g,t,i,j}, V_g)$, where t is the temporal index, (i, j) is the spatial coordinate, and $A = [A_{g,t,i,j}]$ is the output of the action model. $V = [V_g]$ can be either a one-hot vector or with multiple positive entries, depending on the dataset. This weakly-supervised approach is similar to the approach of weakly-supervised object detection in images by Oquab *et al.* [10]. An illustration is shown in Figure 4.1a. An action model trained with this approach will be referred to as a VT model.

In the context of action detection, the supervision provided by such a weakly-supervised approach may be poor, because the musician might not be playing the instrument all the time throughout the video. Furthermore, the action model needs to search through the entire scene for playing actions while there are potentially many movements in the scene that are irrelevant to the playing actions.

A plausible way to improve the performance is to include information from different modalities. Similar to how Simonyan *et al.* fused information from a still-image stream and an action stream in their two-stream action detection model [22], we can fuse information from a still-image stream, an action stream, and an audio (sound) stream so that we have more information available. This could improve the performance, as we will show in Section 4.3.4, but it does not deal with the problem of lacking supervisions itself.

Can we limit the search space for the action model and improve the supervision we have? We can achieve it by using the observation 'action-in-object', stating that the actions responsible to the instrument sound should occur in the region of the instrument. Assume we have a well-trained object model that can inform us the locations of instruments in the form of an output tensor $O = [O_{g,t,i,j}]$ whose value at a given location, (i, j), specifying the confidence level of observing an instrument g there. By binarizing O with respect to

doi:10.6342/NTU201801365

a threshold u, that is,

$$\hat{O}^u_{g,t,i,j} = \begin{cases} 1 & \text{if } O_{g,t,i,j} \geq u \\ 0 & \text{if } O_{g,t,i,j} < u \end{cases}, \tag{4.1}$$

we can limit the search space to the region of the instrument by asking the action model to regard only the region of the instrument as positive, i.e., $loss(A_{g,t,i,j}, \hat{O}^u_{g,t,i,j})$. An illustration is shown in Figure 4.1b. A downside of this approach, however, is that the musician could just bring the instrument alongside without playing it. The appearance of the instrument does not necessarily imply the playing of the instrument at a given time. We will refer to an action model trained with this approach as an OT model.

Table 4.1: Architecture of the sound model. It is an FCN adapted from the model proposed by the authors in their previous work [1]. There are three scales of input feature maps, and each of them has their own stack of early convolutions (Conv1 and Conv2). 'RF' represents receptive field and 'St' represents stride size.

Input ($\times$3 scales)		128 channels Log mel-spectrogram
Early convolutions	Conv1 ($\times$3 scales)	Filter #: 256, RF: 5, Pad: 2, St: 1 Batch norm, Pool: 4
	Conv2 ($\times$3 scales)	Filter #: 256, RF: 5, Pad: 2, St: 1 Batch norm, Pool: 4
		Concatenate 3 scales
Late convolutions	Conv3	Filter #: 512, RF: 1, Pad: 0, St: 1 Batch norm, Dropout
	Conv4	Filter #: 512, RF: 1, Pad: 0, St: 1 Batch norm, Dropout
	Conv5 (S)	Filter #: 9, RF: 1, Pad: 0, St: 1 Sigmoid function
Global pooling		Average pooling

Interestingly, we do have a way to acquire information about when an instrument is played. We can utilize the second observation 'action-making-sound' introduced in Section 1.3 and the beginning of this chapter, stating that an action of instrument playing would usually produce the sound of that instrument. Assume we have a well-trained sound model that can inform us if there are sounds of a specific instrument in a frame in the form of an output matrix $S = [S_{g,t}]$ specifying the confidence level of observing sounds of an

doi:10.6342/NTU201801365

Table 4.2: Architecture of the action model and the object model. It is an FCN adapted from the CNN model VGG_CNN_M_2048 used in [2]. 'RF' represents receptive field and 'St' represents stride size, and 'LRN' represents local response normalization

Input		Object 3 channels	Action 10 channels
Early convolutions	Conv1	Filter #: 96, RF: 7x7, Pad: 3, St: 2 LRN, Pool: 2	
	Conv2	Filter #: 256, RF: 5x5, Pad: 2, St: 2 Pool: 2	
	Conv3	Filter #: 512, RF: 3x3, Pad: 1, St: 1	
	Conv4	Filter #: 512, RF: 3x3, Pad: 1, St: 1	
	Conv5	Filter #: 512, RF: 3x3, Pad: 1, St: 1 Pool: 2	
Late convolutions	Conv6	Filter #: 2048, RF: 3x3, Pad: 0, St: 1 Dropout	
	Conv7	Filter #: 1024, RF: 1x1, Pad: 0, St: 1 Dropout	
	Conv8 (O, A)	Filter #: 9, RF: 1x1, Pad: 0, St: 1 Sigmoid	
Global pooling		Max pooling or None	

doi:10.6342/NTU201801365

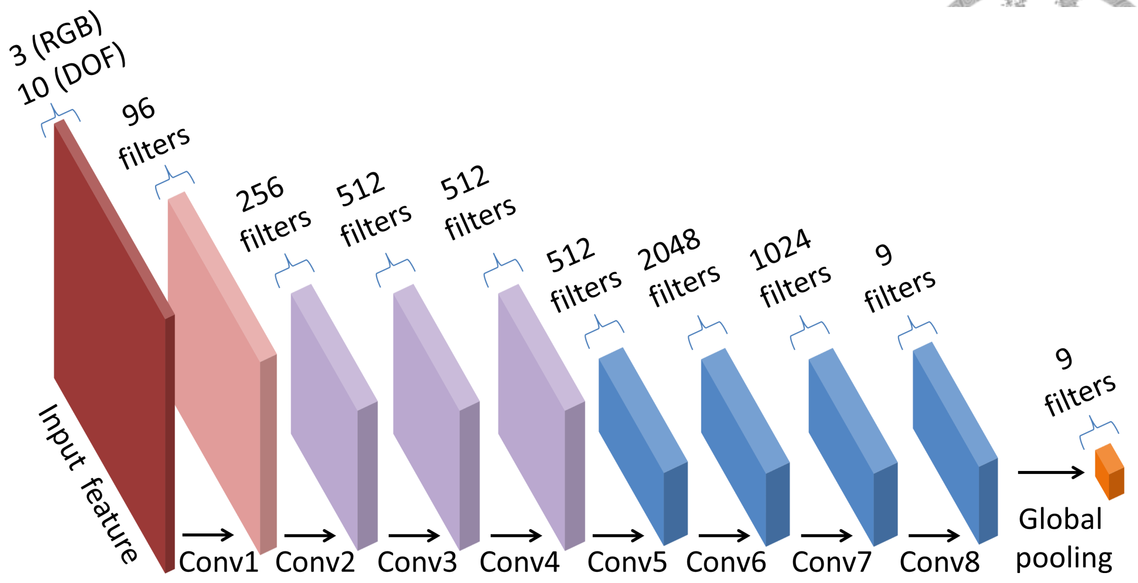


Figure 4.3: Architecture of the action and object models. Details of the model architecture are described in TABLE 4.2.

instrument g at a time frame t. By binarizing S with respect to a threshold v, that is,

$$\hat{S}^{v}_{g,t} = \begin{cases} 1 & \text{if } S_{g,t} \geq v \\ 0 & \text{if } S_{g,t} < v \end{cases}, \tag{4.2}$$

we can inform the action model to consider a frame as containing playing actions of an instrument only if this frame has sounds of the instrument. The corresponding loss for training is $loss(\max_{i,j} A_{g,t,i,j}, \hat{S}^{v}_{g,t})$. An illustration is shown in Figure 4.2a. An action model trained with this approach will be referred to as an ST model.

In order to have the benefits of both an OT model and an ST model, we can combine the output of the object model and the output of the sound model by point-wise multiplication, that is, $\hat{S}^{v}_{g,t}\hat{O}^{u}_{g,t,i,j}$. The corresponding training loss is $loss(A_{g,t,i,j}, \hat{S}^{v}_{g,t}\hat{O}^{u}_{g,t,i,j})$. An illustration is shown in Figure 4.2b. In this way, we inform an action model to search for playing actions only in the region containing the instrument in a frame containing the sounds of the instrument. We will refer to an action model trained with this approach as an SOT model.

In summary, to circumvent the lack of annotated data, our proposed framework tries to acquire supervisions from other modalities. A trained object model provides spatial

doi:10.6342/NTU201801365

supervisions while a trained sound model provides temporal supervisions. Furthermore, we do not need the original clip-level tags anymore when we train an SOT action model. A list of the models used in this paper is presented in Table 4.4.

4.1.3 Fusion of different modality streams after model training

In Section 4.1.2, we have intentionally avoided including either still images or audio information into the input features of the action model so that the action model can be a standalone model without information from other modalities. However, we can still utilize the trained sound and object models to assist the trained action model when they are available.

We adopt a straightforward fusion strategy by point-wisely multiplying the output layers of the models. Similar to the four levels of supervisions introduced in Section 4.1.2, we can also have four levels of assistance from the sound and object models.

Action only $[M_{g,t,i,j}] = [A_{g,t,i,j}]$

Action$\times$Object $[M_{g,t,i,j}] = [A_{g,t,i,j} O_{g,t,i,j}]$

Action$\times$Sound $[M_{g,t,i,j}] = [A_{g,t,i,j} S_{g,t}]$

Action$\times$Object$\times$Sound $[M_{g,t,i,j}] = [A_{g,t,i,j} O_{g,t,i,j} S_{g,t}]$

When we use the last fusion formula, it is equivalent to saying that a playing action of a given instrument g occurs at location (i, j) at time t if there are playing movements of the instrument g at location (i, j) at time t AND there are the appearances of the instrument g at location (i, j) at time t AND there are sounds of the instrument g at time t.

doi:10.6342/NTU201801365

4.2 Experimental setup

4.2.1 Models

Sound model: FCN trained with audios

The sound model produces frame-level instrument sound predictions. Equation (4.2) is used to binarize the raw output, which is then used as part of the target to train the action model.

A fully-convolutional network is implemented as the sound model that performs 1D convolutions. It is similar to the one proposed in Chapter 3 and in my previous work [1], which has led to state-of-the-art result in frame-level music auto-tagging. The sound model used in this chapter differs from the one used in Chapter 3 mainly in four ways. First of all, a different number of filters is used for the early convolution layers, which slightly improves the performance. Second, three instead of six feature maps are used for efficiency. Third, the Gaussian filters are not used in the output because we want a sharper prediction. We find that the Gaussian filters could improve the frame-level predictions but will also blur the boundaries. Fourth, a batch-normalization layer [95] is used after every convolution layer except for the output layer, to make the training process more stable with respect to parameter initialization. Each scale of the input feature maps is processed by its own early convolution layers, and then the outputs from three Conv2 layers are concatenated. The architecture is shown in Table 4.1. We will use the output of the Conv5 layer as the output, S, of a sound model.

Object model

The object model is implemented with an FCN to locate the instruments in the scene. Convolution layers process input feature maps locally so the location information is maintained. Therefore, an FCN can be used to locate objects in images, as proposed by Oquab *et al.* [10]. It takes still RGB images as the input. We will use an architecture similar to VGG_CNN_M_2048 [2] but with some modifications, shown in Table 4.2 and Figure 4.3. Importantly, all the fully-connected layers are replaced with convolution layers. We will

doi:10.6342/NTU201801365

use the output of the Conv8 layer as the output, O, of an object model.

It has been shown that utilizing the pre-trained parameters of a good model can improve the performance for image recognition [22, 79]. Following this light, the object model is modified from the VGG_CNN_M_2048[1] in [2]. Although there are other models that have better accuracy, I choose this one for it has reasonably good performance and for it can fit into the GPU of our computing machine.

One possible way to use the pre-trained CNN model is to directly transform the fully-connected layers in the CNN to convolution layers, as is done by Long *et al.* [79]. However, we have found that this direct conversion results in a poor localization model because, in the original setting of VGG_CNN_M_2048, it takes an image of size 224x224 as the input and produces an output of size 6x6 right before the fully-connected layers, which is too coarse for the purpose of object localization. To cope with this issue, we use only the convolution layers from VGG_CNN_M_2048 and discard all the fully-connected layers. We then append three convolution layers with a smaller 3×3 receptive field as shown in Table 4.2. We train the object model by regarding all the frames in a video clip of an instrument label as positive instances of the instrument, the same as the VT supervision for the action model in Figure 4.1a.

Action model

The action model is implemented with an FCN to capture the actions of instrument-playing. It takes a stack of dense optical flows [22, 80] as the input. We expect the action model to locate the playing actions in the scene. The architecture is the same as the object model but with a different number of input channels. It is trained from scratch without pre-training. We use the output of the Conv8 layer as the output, A, of an action model.

Possible alternative models as the object model

In this work, we want the object model to locate instruments in a frame or an image. There are mainly two types of detection in the literature: semantic/instance segmentation

[1]We use parameters from https://gist.github.com/ksimonyan/78047f3591446d1d7b91

doi:10.6342/NTU201801365

for pixel-level prediction [96] and object detection for bounding-box prediction [97, 98]. In this this work, we prefer the pixel-level prediction because a state-of-the-art model (such as Mask R-CNN [96]) can segment the objects precisely. In contrast, bounding-box could include undesired regions external to the target object. However, a model for pixel-level prediction usually requires training with pixel-level annotations [96] which are expensive to collect. Microsoft COCO [99] is a commonly used dataset for instance segmentation [96]. It contains 91 classes of objects, but contains no instrument classes. Therefore, we are not able to use it to train instance segmentation models.

To address this problem, we may predict the pixel-level labels by using weakly supervised learning as it is done by Oquab *et al.* [10] and as it is done in Section 4.2.1. We may also predict the bounding boxes by using bounding-box annotations, which are easier to collect than the pixel-level annotations. PASCAL VOC datasets are commonly used datasets for bounding-box-based object detection[2] [100], which contain images of 20 classes with bounding-box information, but the classes do not contain instruments either. Nevertheless, we can train an object detection model by using ImageNet-Instrument data we have collected from the ImageNet website, which have been used to evaluate the object model as described in Section 4.2.3.

In Section 4.3.3, we will investigate using Faster R-CNN as an object model for training OT and SOT models.

4.2.2 Features

A sampling rate of 16,000 and hop size of 512 are used to extract log mel-spectrograms from audios, and 3 scales of log mel-spectrograms with window size 512, 2048, and 8192 are used, similar to what is done in Chapter 3. Therefore, the input temporal resolution is $16000/512 = 31.25$ frame-per-second (FPS) in the input feature maps. After the processing of the sound model with 16 total strides, the output has a temporal resolution of $31.25/16 = 1.95$ FPS. We also use this resolution as the temporal resolution for the action and object models. The log mel-spectrograms are extracted with Librosa, an open-source

[2]http://host.robots.ox.ac.uk/pascal/VOC/index.html

doi:10.6342/NTU201801365

Python library for audio analysis [101]. All the images and videos are resized so that the longer side has 256 pixels, maintaining the aspect ratio.

The RGB images are used in the object model. They are sampled from a video clip with 1.95 FPS, which is the same as the temporal resolution of the sound model. Because the FCNs can handle input of arbitrary sizes, we do not have to pad the images.

The dense optical flows are used in the action model. We extract the dense optical flows also with 1.95 FPS. For each frame, we use a stack of five optical flows as the representation, including the dense optical flow of the frame itself and the dense optical flows of its four neighboring frames (two after and two before). Each dense optical flow is decomposed into an x-direction flow and an y-direction flow, so there are totally $5 \times 2 = 10$ channels for the input. We will test five temporal resolutions for the extraction of dense optical flows in Section 4.3.3. To extract the dense optical flows, we convert RGB images to the gray scale and then employ OpenCV[3].

4.2.3 Datasets

We use five datasets in this paper. For training the action and object models, we use a subset of YouTube-8M[4] [102]. For training the sound model, we use the AudioSet[5] [103]. For evaluating the action models, I manually annotate action key points in video clips from 135 videos of YouTube-8M. For evaluating the object model, we collected a set of instrument images from ImageNet. For evaluating the sound model, we use MedleyDB [18]. A list of used datasets can be found in Table 4.3. In this chapter, we focus on the detection of nine instruments. The properties of the instruments and the number of data in the datasets we use are presented in Table 4.5.

YouTube-8M

We use a subset of YouTube-8M dataset [102]. We collected videos for nine instruments according to the tag information provided by YouTube-8M. The nine instruments

[3]http://opencv.org
[4]https://research.google.com/YouTube-8M/
[5]https://research.google.com/audioset/
[6]http://www.image-net.org/

doi:10.6342/NTU201801365

Table 4.3: Datasets used in this paper and their data types, annotation types, and usages. YT8M-IPA represents the proposed YouTube-8M-Instrument-Playing-Action dataset.

Dataset	Data type	Annotation type	Usage in this paper
YouTube-8M [102]	YouTube video	Video-level label	Training of the sound, object, and action models
YT8M-IPA	YouTube video	Playing-action key point I annotate	Evaluation of the action model
AudioSet [103]	YouTube video	Video-level sound label	Training of the sound model
MedleyDB [18]	Music audio	Frame-level sound label	Evaluation of the sound model
ImageNet-Instrument[6]	Image	Instrument bounding box	Evaluation of the object model
MagnaTagATune [17]	Music audio	Track-level music-related label	Training of the sound model

Table 4.4: Models used in this paper and their input features, prediction types, and training targets.

Model name	Input feature	Prediction type	Training target
Sound model	Log mel-spectrogram	Sound activation	Video-level instrument label
Object model	RGB image	Object activation	Video-level instrument label
Video tag as target (VT)	Dense optical flow	Action activation	Video-level instrument label
Sound as target (ST)	Dense optical flow	Action activation	Sound activation
Object as target (OT)	Dense optical flow	Action activation	Object activation
Sound$\times$Object as target (SOT)	Dense optical flow	Action activation	Sound activation$\times$Object activation

doi:10.6342/NTU201801365

are 'Accordion', 'Cello', 'Drummer', 'Flute', 'Guitar', 'Piano', 'Saxophone', 'Trumpet', and 'Violin'. [7]

Note that 'Drummer' is chosen instead of 'Drum' because the 'Drummer' tag seems to contain more instrument-playing videos. We will refer to 'Drummer' tag as 'Drum' in the rest of this work. In addition, I observe that the videos labeled with 'Trumpet' in YouTube-8M contain not only videos of trumpets, but also videos of other instruments in the brass family, such as cornet, French horn, and trombone Therefore, we will treat the 'Trumpet' as a more general trumpet-like tag.

YouTube-8M has divided the data into 'train,' 'validate,' and 'test' sets. 16,804 videos are collected as the training set from YouTube-8M 'train' set, and 2,100 videos as the validation set from YouTube-8M 'validate' set. The first minute in each video clip is used for training. Each instrument has at least 2,000 videos for training. Note that we only need clip-level labels for the training and validation sets.

YouTube-8M-Instrument-Playing-Action

There are no action annotations in YouTube-8M, so we manually annotate a set of video clips from YouTube-8M. The metadata and video IDs of the YouTube-8M 'test' set are not available, so we choose the testing data from YouTube-8M 'validate' set, not overlapping with our validation set. 15 videos are chosen for each instrument. We manually annotate frames in the 0 to 10 seconds and 30 to 40 seconds so that we can evaluate the performance of our model for action detection. With the temporal resolution 1.95 FPS, this comprises 5,400 snapshots. This set of annotations is used only for evaluating action models, not for training. We will refer to this subset with manual annotations as YouTube-8M-Instrument-Playing-Action, or YT8M-IPA for short.

The locations of instrument-playing actions are represented as key points, instead of regions that are commonly used in the literature of action detection [19, 74]. I choose to

[7]While there are certainly other instruments, we choose these nine instruments mainly for they cover instrument types that are commonly seen. On one hand, we have sufficient number of training data for each of them in the datasets we use. On the other hand, we still need to manually annotate the action locations of the chosen instruments for evaluation (because such labels are not available elsewhere) so we have to limit the number of instruments. As the proposed methodology is quite generic, we believe our model can be easily extended to deal with other instruments in the future work.

doi:10.6342/NTU201801365

Table 4.5: Properties of the nine instruments. The lower part of the table contains the number of data in the datasets. In YT8M-IPA, the playing actions are annotated at the intersections of the action regions and the playing tools as described in Section 4.2.3. ‘frs’ represents frames. ‘imgs’ represents images.

	Accordion	Cello	Drum	Flute	Guitar	Piano	Saxophone	Trumpet	Violin
Action region	Keys/body	Strings	Drum skins	Holes/mouthpiece	Strings	Keys	Keys/mouthpiece	Valves/mouthpiece	Strings
Playing tool	Hands	Hand/bow	Sticks	Hands/mouth	Hands	Hands	Hands/mouth	Hands/mouth	Hand/bow
Portable?	✓	✓	×	✓	✓	×	✓	✓	✓
YouTube-8M (clips)	2279	2260	2495	2264	3204	3678	2367	2240	3521
YT8M-IPA (clips/frs)	15/600	15/600	15/600	15/600	15/600	15/600	15/600	15/600	15/600
AudioSet (clips)	2658	4664	4866	4281	5624	5233	2966	3654	6553
MedleyDB (songs)	5	11	65	10	64	43	5	7	14
ImageNet-Inst. (imgs)	412	323	252	359	135	315	343	294	365

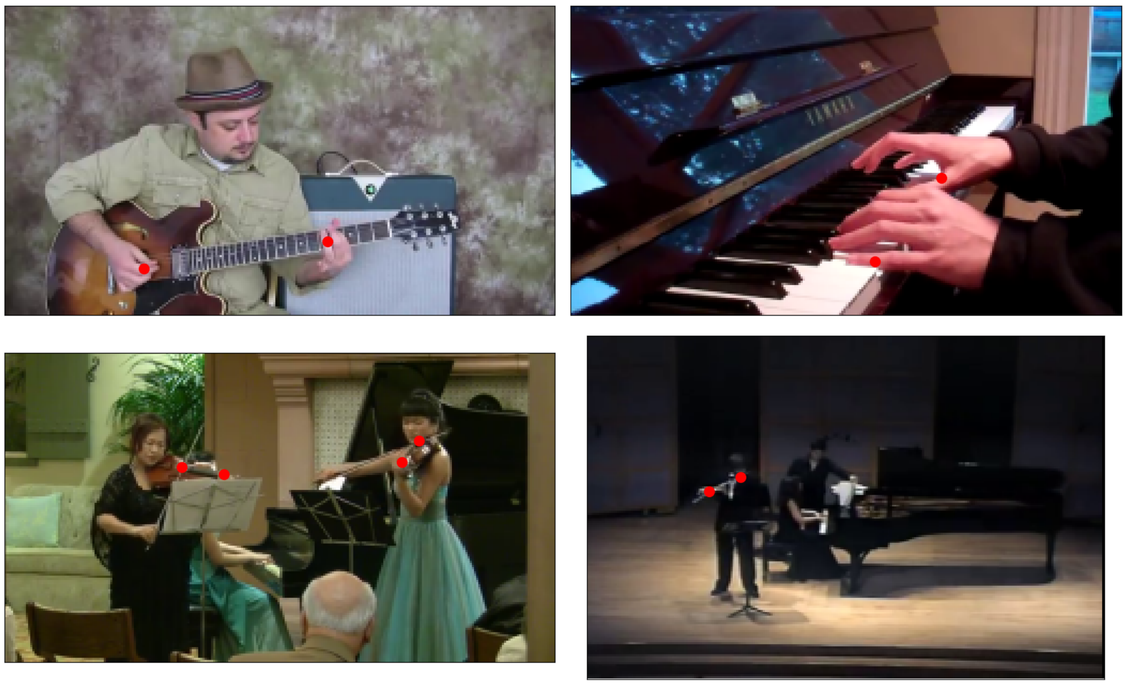


Figure 4.4: Examples of the manually annotated key points (red dots) of instrument-playing actions. I annotate the locations that are most directly responsible to making the instrument sounds as described in Section 4.2.3. Best seen in color.

do so because I want to predict the actions that are most directly responsible for making instrument sounds. The sounds are usually made by the contacts between sound making tools, such as hands and sticks, and an instrument, and the contacts are usually more like points than regions. Some examples of the annotations are shown in Figure 4.4.

The author annotates the locations of the instrument playing according to the following principles. For wind instruments like flute, saxophone, and trumpet, the locations where the hands are pressing and the location of the mouth are labeled. For string instruments like cello, violin, and guitar, the location of the pressing hand and the intersection of the stroking hand (or the bow) and the strings are labeled. For accordion, the two hands are labeled, and the center of accordion is also labeled because the deformation of the accordion is also an indicator of playing. For drum and piano, the locations of the hands/sticks hitting the instruments are labeled. Note that these locations are labeled only if the instruments in sight are responsible for making the sounds at a given frame.

doi:10.6342/NTU201801365

Table 4.6: Evaluation of the sound models for instrument sound detection. The sound model trained with AudioSet outperforms the one trained with YouTube-8M and the one (the model in our previous work [1]) trained with the music dataset MagnaTagATune. We use 'Acc.' as shorthand for Accordion.

	MagnaTagATune (MTT)	Subset of YouTube-8M (YT8M)	Subset of AudioSet (AS)	MTT+YT8M+AS
Accordion	NA	0.806	0.714	0.724
Cello	0.877	0.805	0.798	0.871
Drum	0.805	0.705	0.773	0.763
Flute	0.752	0.803	0.79	0.802
Guitar	0.765	0.74	0.735	0.771
Piano	0.688	0.779	0.75	0.761
Saxophone	0.717	0.747	0.91	0.86
Trumpet	0.644	0.911	0.92	0.886
Violin	0.871	0.878	0.821	0.855
AVG	NA	0.797	0.801	0.81
AVG w/o Acc.	0.765	0.796	0.812	0.821

ImageNet-Instrument

In order to evaluate the object localization ability of the object model, images of the nine instruments were collected from the ImageNet website. They also provide the bounding boxes for the locations of the instruments. Totally 2,798 images and the corresponding bounding boxes are collected. An instrument has on average 311 images ranging from 135 to 412.

MedleyDB

MedleyDB [18] is a multi-track instrument dataset. It also contains the timestamps of the occurrences of the instrument sounds. There are totally 111 songs with the nine instruments used in this work.[8] We use it to evaluate the sound model.

[8]We aggregate the 'acoustic guitar,' 'clean electric guitar,' and 'distorted electric guitar' in MedleyDB into the 'Guitar' tag, and aggregate 'baritone saxophone,' 'soprano saxophone,' and 'tenor saxophone' in MedleyDB into the 'Saxophone' tag.

doi:10.6342/NTU201801365

AudioSet

AudioSet [103] is a video dataset released by Google, containing audio annotations on a 10-second clip in each of the videos. A subset of the nine instruments is collected from AudioSet, consisting of 35,512 video clips for training and 902 video clips for validation. Each instrument has 3,945 training clips on average.

4.2.4 Training

For training the action and object models, we use the first 60 seconds in each training video clip from YouTube-8M as the training data. The 60-second video clip is divided into 12 consecutive 5-second sub-clips. Under the resolution of 1.95 FPS, each sub-video contains 10 frames. For each mini-batch, a video is randomly picked without replacement and then a sub-clip is randomly picked from the 12 sub-clips of the video, so the size of a mini-batch is 10. We follow Simonyan *et al.* [22] to use stochastic gradient descent with 0.9 momentum, but fix the learning rate to 0.001. For training the sound model, the 10-second annotated sub-clip from AudioSet is used, and AdaGrad [93] with 0.01 initial learning rate is used to adjust the learning rate as done in Chapter 3. The loss function for all models is the binary cross-entropy, $-(q\log(p) + (1-q)\log(1-p))$, where q is the target and p is the output of a model.

We find that directly training the object model with pre-trained parameters would not perform well, even worse than training from scratch without pre-trained parameters. The loss almost never decreases. This could be due to the mismatch of scales in the pre-trained parameters in the early convolutions and the randomly sampled parameters in the late convolutions, which makes the back-propagation difficult. Therefore, we adopt a two-step training process. First we freeze the parameters in the early convolutions and only allow updating the parameters in the late convolutions. After 20 epochs of training, we free the parameters in the early convolutions so that they start to update for 30 more epochs. For the sound model, 100 epochs are executed. For the action models, 100 epochs are executed for the experiments in Section 4.3.3. The remaining experiments thereafter will use the best one in Section 4.3.3 as a pre-trained model and train for another 30 epochs.

doi:10.6342/NTU201801365

The parameters from the epoch with the best clip-level AUC[9] are picked as the parameters for testing. I implement the models with PyTorch[10].

4.3 Experiments

The experiments are reported in this section. We will first evaluate independently the performance of the sound model (for instrument sound detection) and the object model (for instrument object detection). In the evaluation of the action model (for playing action detection), we will first see the effect of the temporal resolution of dense optical flows, and then we evaluate the performance of the action models trained with the four different targets discussed in Section 4.1.2.

There are three models interact with each other, so we intend to investigate the performance of each model and also how the two auxiliary models affect the performance of the action model. Specifically, we want to answer the following questions:

1. How does the resolution of dense optical flows affect the performance of playing action detection?

2. Do the supervisions provided by the sound and object models improve the performance of playing action detection?

3. How does the effectiveness of the sound model affect the effectiveness of the ST and SOT action models?

4. Does the fusion with different streams of modalities (sound and object) improve the performance of playing action detection?

[9]We do not use frame-level AUC here for we do not have frame-level annotations at all for the training and validation sets. AUC stands for Area Under the Curve, a widely used metric for classification problems.

[10]`http://pytorch.org`

doi:10.6342/NTU201801365

4.3.1 Performance of the sound model for instrument sound detection

We evaluate the ability of temporal localization of the sound model with the music dataset MedleyDB. We compute the per-class AUC by taking each frame as an instance and computing the score over all frames.

Table 4.7: Evaluation of the object models for instrument object detection. They are tested on the instrument images with bounding boxes collected from ImageNet (HIT RATE). The one initialized with a pre-trained model VGG_CNN_M_2048 (With pre-training) outperforms the one with random initialization (From scratch) by a large margin.

	From scratch	With pre-training
Accordion object	0.845	0.927
Cello object	0.74	0.916
Drum object	0.54	0.579
Flute object	0.345	0.671
Guitar object	0.719	0.963
Piano object	0.841	0.946
Saxophone object	0.77	0.945
Trumpet object	0.507	0.833
Violin object	0.627	0.847
Average	0.659	0.848

We compare three sound models. The first one is trained with YouTube-8M and the architecture in Table 4.1. The second is trained with AudioSet and the same architecture. The third model is the one presented in our previous work [1][11], which was trained with the music dataset MagnaTagATune [17]. The output of the third model covers eight out of the nine instruments used in this paper except for accordion, so we compute the AUCs of these eight instruments for this model. The result is shown in Table 4.6.

In general, when the model is trained with a single dataset, the best result is obtained by the model trained with AudioSet, achieving 0.801 AUC, and 0.812 AUC excluding the Accordion sound. This can be expected because AudioSet has better annotations. The YouTube-8M sound model is close to the AudioSet sound model, achieving 0.797

[11] https://github.com/ciaua/clip2frame/blob/master/data/models/model.20160309_111546.npz

doi:10.6342/NTU201801365

AUC (0.796 AUC excluding the Accordion sound). Although the MagnaTagATune sound model is in general inferior to the other two models, it performs well in some instrument sounds, such as the Cello sound, the Drum sound, and the Violin sound.

An additional model is also trained with the data combining MagnaTagATune, YouTube-8M, and AudioSet. It outperforms the models trained with a single dataset, achieving 0.810 AUC, and 0.821 AUC excluding Accordion.

Although the model trained with the data combining the three datasets achieves the best performance when it is evaluated with MedleyDB, we will still mainly use the AudioSet sound model as the sound model in the following experiments for two reasons. First, we can observe that ST action models trained with an AudioSet sound model will yield better performance than those trained with an MTT+YT8M+AudioSet sound model, as will be shown in Section 4.3.3. Second, I conceive it better to use only one dataset for training, if possible, for the sake of simplicity. In addition, as the AudioSet actually covers many more instruments and other action types, in the future one can use the same data source for extension.

4.3.2 Performance of the object model for instrument object detection

We evaluate the ability of spatial localization of the object model independently with ImageNet instrument images. ImageNet website provides the bounding boxes of the locations of the instruments. We compute the accuracy in a way similar to Oquab *et al.* [10]. Each instrument is processed separately. For an image in an instrument class, the prediction of the object is considered as a hit if the location of the maximum prediction value is inside the bounding box.

The result is shown in Table 4.7. The model with pre-training achieves 0.848 on average, which is greatly higher than the 0.659 obtained by the model trained from scratch. We refer to the model with pre-training as the object model hereafter. Most instruments have hit rate higher than 80%. The Flute object is one of the most challenging cases due to its thin body. The Drum object achieves the worst hit rate, 0.579. By looking into the

doi:10.6342/NTU201801365

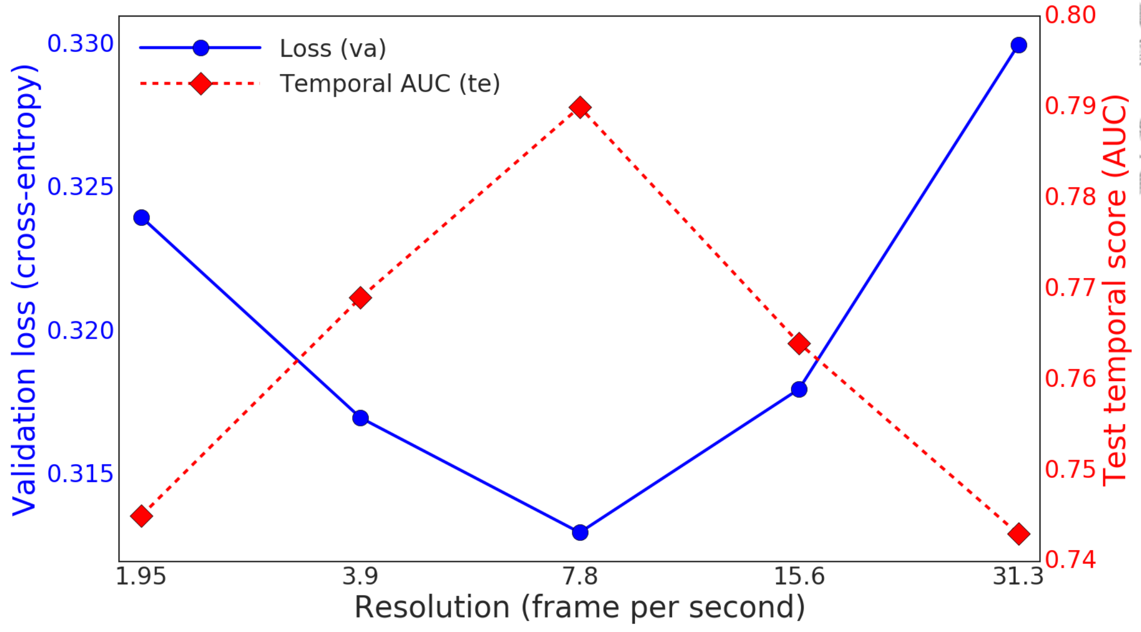


Figure 4.5: Effect of different temporal resolutions of dense optical flows. The blue solid line and the red dashed line are the validation losses and the test temporal AUC, respectively, of the action models trained with different resolutions of dense optical flows. The validation loss (the lower the better) and the test AUC (the higher the better) are both optimal at the resolution of 7.8 frame-per-second.

images collected from ImageNet, we find that the images include various types of drums. As far as we have seen, most videos with 'Drummer' tag in YouTube-8M have jazz drum set. Therefore, the object model trained with them could perform worse for other types of drums.

4.3.3 Performance of the action model for instrument-playing action detection

We evaluate the temporal and spatial accuracy of the proposed models. I have manually constructed the test set by annotating the locations that are relevant to the playing actions, as mentioned in Section 4.2. For the temporal evaluation, a frame is a positive instance if the set of annotated locations of the frame is non-empty, and a negative instance otherwise. We evaluate each instrument separately with AUC. The AUC is first computed on the frames within a video clip and then the scores are averaged over all the video clips.

For spatial evaluation, the evaluation is performed in a way similar to the evaluation of the object model by evaluating the location of the maximum prediction value in a frame,

doi:10.6342/NTU201801365

referred to as the *max location*. However, the action locations are represented as coordinates instead of bounding boxes, so we compute the minimum distance between the max location and the annotated locations, that is,

$$D = \min_{l \in L} \|l - M\|_2, \tag{4.3}$$

where L is the set of the coordinates of the annotated locations in a frame, and M is the coordinate of the max location.[12]

Effect of resolutions of dense optical flows

The temporal resolution will affect how subtle the movements the model can see, so the choice of the temporal resolution of dense optical flows for the action model is investigated. A stack of 5 consecutive dense optical flows is used. Five resolutions are tested that are multiples of the base temporal resolution, 1.95, 3.9, 7.8, 15.6, and 31.3 FPS. The last one is roughly the default FPS of most online videos.

Let's assume for a moment that we are using a resolution of 2 FPS. A stack of 5 frames cover about $5 \times (1/2) = 2.5$ second. If we use a finer resolution, say 8 FPS, the stack will cover $5 \times (1/8) = 0.625$ second. The movements of instrument-playing actions are not always fast, so we will desire the feature to cover a reasonably span of time that is long enough to capture the playing movements. The simplest way to cover a larger period of time is to use a larger stack of dense optical flows, say 20 consecutive frames, but this will also largely increase the computational time and resource. To keep the experiments manageable, we will maintain a stack of 5 frames and only vary the temporal resolutions.

We measure the effectiveness with the validation loss and also the temporal accuracy on the test set. The result is shown in Figure 4.5. We can see that both measures favor the resolution of 7.8 FPS. Therefore, we will use this resolution in all the experiments that follow. A stack of 5 consecutive dense optical flows with this resolution will cover 0.64

[12]We note that intersection-over-union (IoU) is a metric that is commonly used in action detection or object detection that involves prediction of regions [19,74,97,98,104], in which actions are attributed to the entire object in action, such as a human being or an animal. We opt for not using IoU in our work, because the locations of the instrument-playing actions we consider in this work are composed of very small parts and, hence, it is more suitable to represent them as key points, as described in Section 4.2.3.

doi:10.6342/NTU201801365

Table 4.8: Evaluation of the action models for instrument-playing action detection. The ST and SOT models outperform other models temporally because they have temporal supervisions from the sound model, while the OT and SOT models outperform other models spatially because they have spatial supervisions from the object model.

Model	Threshold		Score	
	Sound	Object	Temporal (AUC)	Spatial (pixel)
Center	NA	NA	NA	37.2
Sound model	NA	NA	0.881	NA
Object model	NA	NA	0.712	33.3
Sound × Object	NA	NA	0.900	33.3
Video tag as target (VT)	NA	NA	0.790	33.9
Sound as target (ST01)	0.1	NA	0.824	32.7
Sound as target (ST03)	0.3	NA	0.827	32.2
Sound as target (ST05)	0.5	NA	0.827	33.7
Sound as target (ST07)	0.7	NA	0.830	33.0
Sound as target (ST09)	0.9	NA	0.824	33.7
Object as target (OT01)	NA	0.1	0.680	45.8
Object as target (OT03)	NA	0.3	0.809	29.4
Object as target (OT05)	NA	0.5	0.809	29.5
Object as target (OT07)	NA	0.7	0.798	29.2
Object as target (OT09)	NA	0.9	0.788	31.0
S×O as target (SOT0501)	0.5	0.1	0.828	29.4
S×O as target (SOT0503)	0.5	0.3	**0.834**	29.1
S×O as target (SOT0505)	0.5	0.5	0.827	**28.9**
S×O as target (SOT0507)	0.5	0.7	0.820	29.5
S×O as target (SOT0509)	0.5	0.9	0.803	30.7

doi:10.6342/NTU201801365

Table 4.9: Instrument-wise action detection performance (Temporal in AUC). The best temporal scores are all achieved by ST and SOT that have temporal supervisions from the sound model.

Model (thresholds)	Object model	Video tag as target	Sound as target (0.5)	Object as target (0.3)	S×O as target (0.5, 0.3)
Accordion action	0.695	0.856	0.866	0.887	**0.894**
Cello action	0.812	0.918	0.919	0.932	**0.937**
Drum action	0.609	0.824	**0.853**	0.823	0.846
Flute action	0.751	0.729	0.783	0.744	**0.785**
Guitar action	0.723	0.813	**0.871**	0.787	0.838
Piano action	0.582	0.742	0.77	0.697	**0.778**
Saxophone action	0.715	0.666	0.741	0.751	**0.754**
Trumpet action	0.694	0.685	0.757	0.772	**0.777**
Violin action	0.823	0.875	0.881	0.891	**0.899**
Average	0.712	0.79	0.827	0.809	**0.834**

Table 4.10: Instrument-wise action detection performance (Spatial in pixel). The best spatial scores are mostly achieved by OT and SOT that have spatial supervisions from the object model.

Model (thresholds)	Object model	Video tag as target	Sound as target (0.5)	Object as target (0.3)	S×O as target (0.5, 0.3)
Accordion action	27.7	24.9	27.1	24.3	**23.8**
Cello action	**21.8**	28.8	27.2	23.1	25.9
Drum action	56.7	42.2	41.8	45.1	**40.7**
Flute action	27.6	30.5	36.5	27.4	**26.9**
Guitar action	29.5	**28.1**	29.3	29.2	28.4
Piano action	51.1	**31.7**	32.9	39.4	31.9
Saxophone action	25.3	38.3	30.5	**23.2**	26.6
Trumpet action	39.6	56.7	49.8	**34.3**	38.1
Violin action	20.5	23.9	24.5	**19**	19.7
Average	33.3	33.9	33.3	29.4	**29.1**

doi:10.6342/NTU201801365

second.

Effect of different training targets

In this subsection, we evaluate the performance of action models trained with four different targets discussed in Section 4.1.2. In addition to the models trained with the four types of targets, we also include three baseline models. The first one is the object model for both the temporal and the spatial evaluations, that is, predicting the presence of playing actions simply by the presence of the instruments. The second one is the sound model for the temporal evaluation, that is, predicting the presence of playing actions simply by the presence of the instrument sounds. The third one is for the spatial evaluation by always predicting the center of the scene in each frame. It is sometimes a good guess for videos because the cameras are often centered at the player of the main instrument.

The average result over all nine instruments are shown in Table 4.8. The sound model performs very well temporally. It verifies that the sounds are indeed important cues for the playing actions. The object model performs well spatially because the characterizing portion of an instrument is often also the sounding part. Predicting the center is among the worst models. We will use Sound $\times$ Object as the training target, so it is interesting to see what if we directly use them as the action prediction. This result is shown in the 'Sound $\times$ Object' entry in Table 4.8. Similar to using sound model as the action model, it performs very well temporally. It has the same spatial performance as the object model.

We will use abbreviations of the format "<training target><threshold v for the sound model if available><threshold u for the object model if available>" as the name for an action model. For example, OT03 is the action model trained with the target produced by the object model with a threshold $u = 0.3$, and SOT0503 is the action model trained with the target produced by the sound model with a threshold $v = 0.5$ and the object model with a threshold $u = 0.3$. In general, we can see that the ST models outperform the VT model temporally, OT models outperform the VT model spatially, and the SOT models outperforms the VT model both temporally and spatially, when the thresholds are not too extreme, that is, between 0.3 and 0.7.

doi:10.6342/NTU201801365

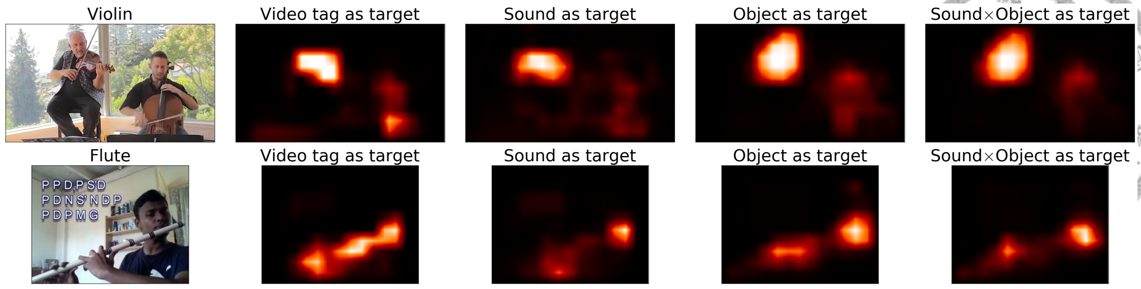


Figure 4.6: The result of instrument-playing action detection as a function of different training targets. As the level of supervision changes from left to right, the result of action detection becomes cleaner and more accurate. The original videos of the two examples are uploaded by Jeremy Cohen (YouTue ID: 2G2VaBX24So) and Krishan Chotoe (YouTube ID: 55_RhFOyRgk), respectively, both of which are under Creative Common license.

Table 4.11: Temporal performance of the ST action models by using as the training target either the sound model trained with the video dataset AudioSet (AS) or the one trained with the music audio dataset MagnaTagATune (MTT). The ST action model using the AudioSet sound model outperforms the one using the MagnaTagATune sound model in all instruments.

Action model (sound threshold)	ST: AS sound (0.5)	ST: MTT sound (0.5)	ST: MTT sound (0.3)	ST: MTT sound (0.1)	ST: MTT+YT8M+AS sound (0.5)
Accordion action	0.866	NA	NA	NA	0.873
Cello action	**0.919**	0.813	0.855	0.825	0.911
Drum action	**0.853**	0.751	0.71	0.822	0.847
Flute action	**0.783**	0.565	0.588	0.583	0.764
Guitar action	0.871	0.718	0.676	0.719	**0.874**
Piano action	**0.77**	0.522	0.498	0.462	0.799
Saxophone action	0.741	0.644	0.558	0.535	**0.748**
Trumpet action	**0.757**	0.547	0.522	0.529	0.704
Violin action	**0.881**	0.799	0.777	0.792	0.872
Average w/o Acc.	**0.822**	0.67	0.648	0.658	0.815

We also perform t-test on some pairs of models with 0.05 confidence level over the averages of all test videos. We found that ST05 is significantly better than VT and all OT models temporally and OT03 is significantly better than VT and all ST models spatially. Similarly, SOT0503 is significantly better than those without temporal (or spatial) supervision temporally (or spatially). The ST models are provided with temporal supervision so the performance is improved most temporally over the VT baseline model, while has less temporal improvement. On the other hand, the OT models are provided with spatial supervision so the performance is improved most spatially over the VT baseline model while have less spatial improvement. The SOT models are provided with both temporal

doi:10.6342/NTU201801365

and spatial supervisions so the performance is improved both temporally and spatially, compared with the VT model. The best temporal score 0.834 AUC and the best spatial score 28.9 pixels are both achieved by the SOT models. Thresholds have some impact on the performance for either ST, OT, or SOT models, but the performance is pretty stable if we choose thresholds around 0.5.

Next, we show the instrument-wise result in Table 4.9 and Table 4.10. The best temporal scores are all achieved by ST and SOT that have temporal supervisions from the sound model. Similarly, the best spatial scores are mostly achieved by OT and SOT that have spatial supervisions from the object model. The instruments in wind family are among the most challenging cases. The object model tends to fail for actions of instruments that are difficult for the player to carry along. For example, we can see that the object model performs poorly in detecting the Drum action and the Piano action, achieving 0.609 and 0.582 AUCs, respectively. For the temporal performance, the inclusion of temporal supervision (ST and SOT models) improves the result for action detection of all the instruments, especially for the instruments in the wind family, Flute, Saxophone, and Trumpet. The movements of the instruments in the wind family are subtle, so the difference of positive and negative frames is small without temporal supervision. For the spatial performance, the object model again performs poorly in detecting the Drum action and the Piano action due to the large sizes of them. The action detections of the wind family instruments Flute, Saxophone, and Trumpet have large gains from the spatial supervision provided by the object model. Cello and Violin actions are also improved.

In general, the supervisions provided by the two auxiliary models significantly improve the performance temporally and spatially. We can also see this visually in Figure 4.6. Comparing with the result of the other three models, the result of SOT is more concentrated and less noisy. Some examples of all nine instruments are shown in Figure 4.7, including some difficult cases.

doi:10.6342/NTU201801365

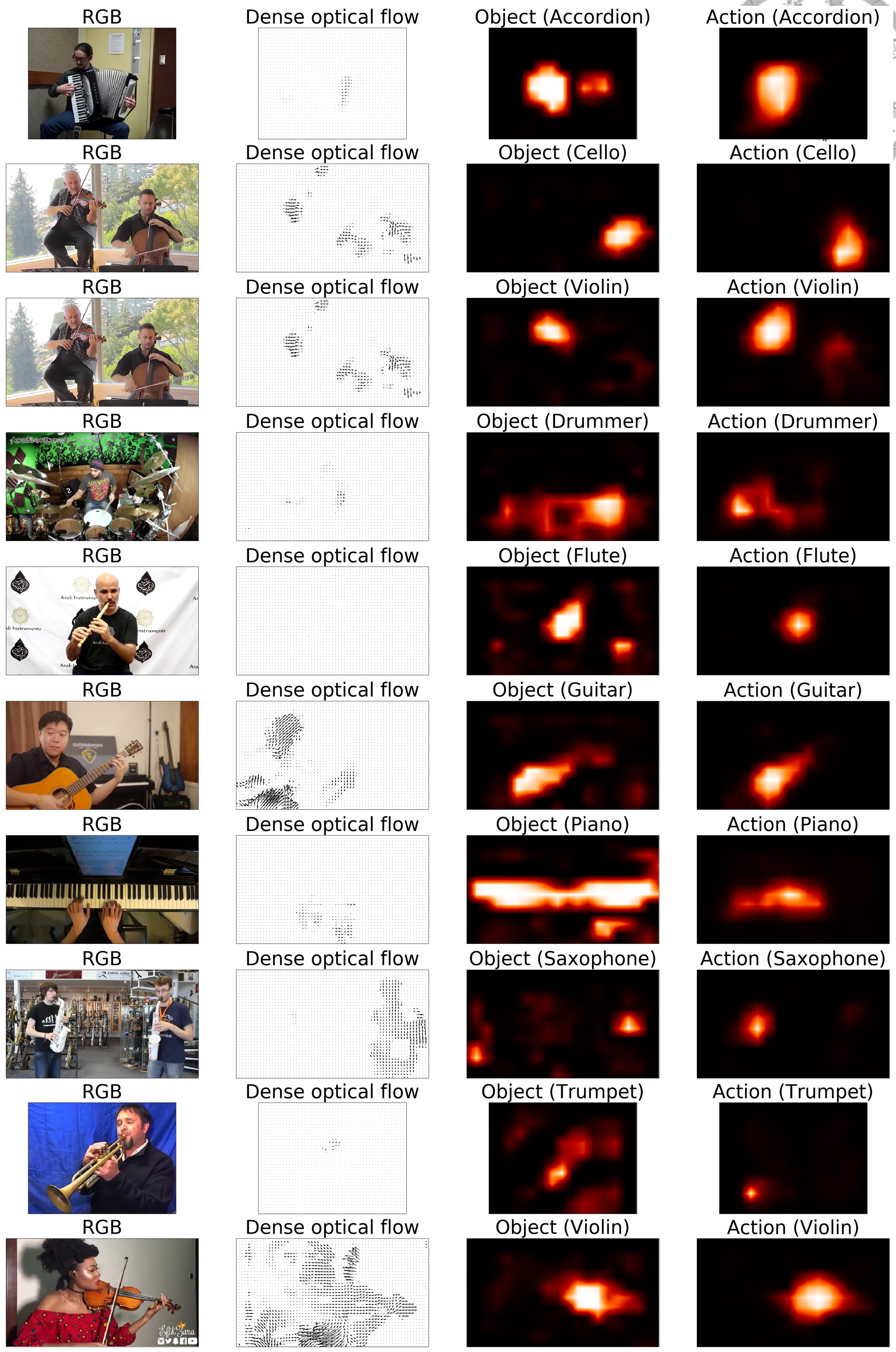


Figure 4.7: Examples of predictions of all nine instruments from the action model SOT0503

doi:10.6342/NTU201801365

Effect of different sound models

In this subsection, we want to see how the performance of the sound model affects the performance of the ST models for action detection. We compare the ST model trained with the AudioSet sound model and the ST model trained with the MagnaTagATune sound model. This comparison is interesting because there are performance gaps between the two models for some instruments in instrument sound detection, as shown in Table 4.6. Additionally, An ST model is also trained with the MTT+YT8M+AS sound model. The result of this experiment is shown in Table 4.11.

First, we observe that the ST models trained with the MTT+YT8M+AS sound model and the AudioSet sound model have close performance. Accordingly, we only consider the AudioSet sound model hereafter for its simplicity.

From the result shown in Table 4.6, we can see that the MagnaTagATune sound model itself performs poorly in the Piano sound, the Saxophone sound, and the Trumpet sound, so it is not surprising that the ST action models trained with MagnaTagATune sound model also perform poorly in the Piano action, the Saxophone action, and the Trumpet action.

Table 4.12: Fusion of different streams of modalities after training (Temporal performance in AUC). The output of an action model is fused with the output of the sound model and/ or the output of the object model by point-wise multiplication. The fusion significantly improves the result.

	SOT (0.5, 0.3)	SOT ×Object	SOT ×Sound	SOT ×Object×Sound	VT ×Object×Sound
Accordion action	0.894	0.895	0.951	**0.954**	0.948
Cello action	0.937	0.945	0.965	**0.974**	0.965
Drum action	0.846	0.847	**0.900**	0.895	0.892
Flute action	0.785	0.799	0.912	**0.921**	0.919
Guitar action	0.838	**0.875**	0.850	0.857	0.846
Piano action	0.778	0.830	0.932	**0.943**	0.942
Saxophone action	0.754	0.767	0.884	**0.888**	0.878
Trumpet action	0.777	0.811	0.888	**0.892**	0.878
Violin action	0.899	0.913	**0.931**	0.929	0.919
Average	0.834	0.854	0.912	**0.917**	0.91

On the other hand, we can see that the MagnaTagATune sound model outperforms the AudioSet sound model in the Cello sound, the Drum sound, the Guitar sound, and the Violin sound when they are evaluated with MedleyDB in Table 4.6. However, the

doi:10.6342/NTU201801365

Table 4.13: Fusion of different streams of modalities after training (Spatial performance in Pixel). The output of an action model is fused with the output of the sound model and/or the output of the object model by point-wise multiplication. The fusion significantly improves the result.

	SOT (0.5, 0.3)	SOT$\times$Object	VT$\times$Object
Accordion action	**23.8**	**23.8**	24.1
Cello action	25.9	**20.2**	20.4
Drum action	40.7	**37.5**	39.2
Flute action	26.9	**22.5**	24.6
Guitar action	28.4	**26.2**	27.1
Piano action	31.9	**27.8**	31.5
Saxophone action	26.6	**20.2**	34.0
Trumpet action	38.1	**31.0**	43.9
Violin action	19.7	**18.6**	18.9
Average	29.1	**25.3**	29.3

ST action models trained with the MagnaTagATune sound model are still inferior to the one trained with the AudioSet sound model in the actions of these instruments. The main reason could be the mismatch of the recording conditions. While the evaluation shown in Table 4.11 may favor the ST model trained with the AudioSet sound model as the action test set (i.e. YouTube-8M) is composed of online videos, the evaluation shown in Table 4.6 may have given the MagnaTagATune sound model some advantages as the sound test set (i.e. MedleyDB) is composed of also high-quality music audios.

In summary, we can see that the quality of the sound model has significant impact on the performance of the action models supervised by them.

Effect of using Faster R-CNN as the object model

In this subsection, we experiment using Faster R-CNN as the object model. We conduct this experiment by modifying the code from https://github.com/ruotianluo/pytorch-faster-rcnn and re-organizing the data of ImageNet-Instrument.

The Faster R-CNN object model achieves 96.8% hit rate if we evaluate it in the way we evaluate the object model in Section 4.3.2. We already use the ImageNet-Instrument data for training the Faster R-CNN model, so this is not a fair comparison. Nonetheless, we also test the Faster R-CNN model on several external images containing instruments and find it performs very well. Some examples of the predicted bounding boxes of the Faster

doi:10.6342/NTU201801365

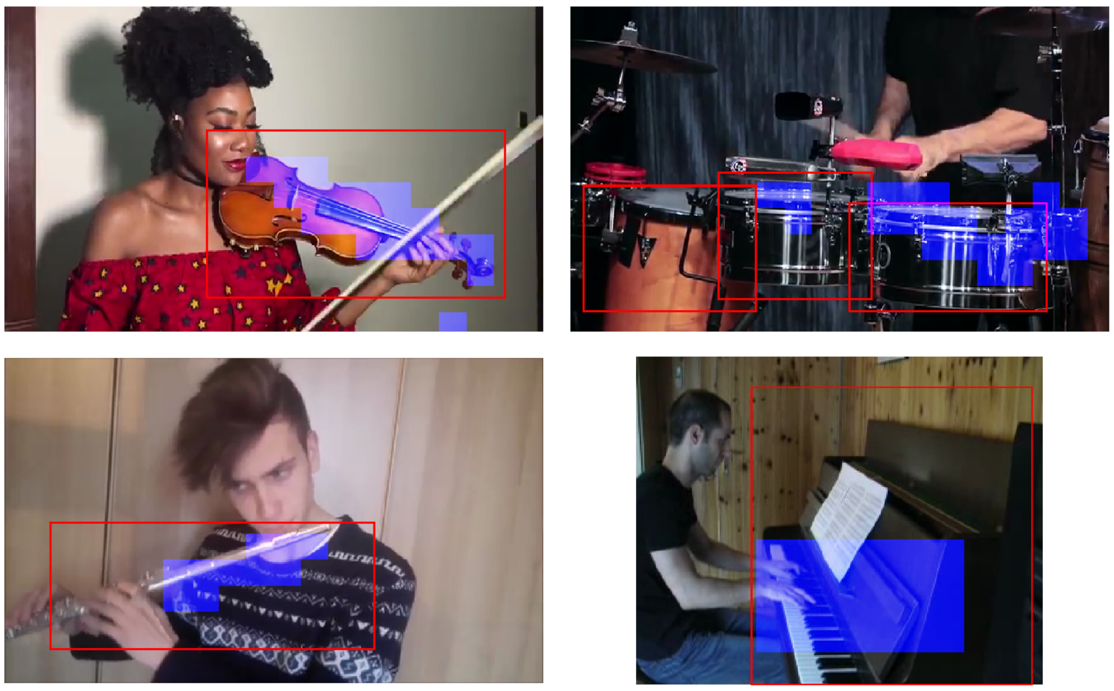


Figure 4.8: Prediction of the weakly-supervised object model used in this paper and the prediction of a Faster R-CNN model. The red rectangle frames are the predictions of the Faster R-CNN model, and the blue shades are the predictions of the weakly-supervised object model. Best seen in color.

R-CNN model are shown in Figure 4.8. The red bounding boxes are the predictions of the Faster R-CNN model, and we show the predictions of the weakly-supervised object model alongside with blue shades. We can see that the Faster R-CNN predictions are pretty accurate, but the bounding boxes also contain large amount of parts that are not instruments or are not relevant to making instrument sounds due to the shape of the bounding boxes. In contrast, the predictions of the weakly-supervised object model often fail to cover the entire instrument, but they fit better the shape of the instruments.

We use the trained Faster R-CNN as the object model and train OT and SOT models by applying five thresholds (0.1, 0.3, 0.5, 0.7, and 0.9) to the class scores of Faster R-CNN. As the weakly-supervised object model and the Faster R-CNN object model are very different models, they may have different optimal thresholds. Therefore, we use their own best threshold for each instrument in order to compare their performance. The result is shown in Table 4.14.

Despite of the good performance of Faster R-CNN in predicting bounding boxes, we find that training SOT and OT action models with it does not yield better performance. In

doi:10.6342/NTU201801365

terms of the temporal performance of playing-action detection, the action models trained with the weakly-supervised object models and Faster R-CNN are close. We can see that the Faster R-CNN object model is better at Drum, Guitar and Piano for the OT model and is better at Drum and Piano for the SOT model. Either OT or SOT gets better result for Drum and Piano with the Faster R-CNN object model. It is interesting that Drum and Piano happen to be the two non-portable instruments among the nine instruments due to their sizes (cf. Table 4.5).

Our conjecture is that this difference in Drum and Piano actions results from the ratio of the playing region of an instrument to the size of the whole instrument. For Drum and Piano, the playing regions are relatively small compared to the size of the entire instruments. As we can see from Figure 4.8, the weakly-supervised object model only predicts as positive a small portion of the piano and drums, which will make it more difficult to intersect with hands or sticks. In contrast, Faster R-CNN can very nicely recognize the entire instrument which will also include relevant body motions other than the playing actions. Therefore, Faster R-CNN could better predict the playing actions temporally in Drum and Piano in comparison to the weakly-supervised object model.

This argument is also supported by the spatial performance of the action models trained with Faster R-CNN. The larger region predicted by Faster R-CNN model will also produce more false positives spatially. Therefore, the average spatial performance of the action models trained with Faster R-CNN is worse than that with the weakly-supervised object model, and it even gets worse in Drum and Piano.

4.3.4 Fusion of different streams after training

In this subsection, we want to see how much the inclusion of other modalities as input information helps the detection by using the fusion scheme described in Section 4.1.3. The sound model and the object model are only used as the training target in the previous subsections, while the predictions of the sound model and the object model are directly used to assist the prediction of the action model in this subsection.

The fusion result is shown in Table 4.12 and Table 4.13. The fusion significantly im-

doi:10.6342/NTU201801365

Table 4.14: The performance of the action models trained with a Faster R-CNN object model in comparison with the action models trained with a weakly-supervised object model. We train the action models with five thresholds in the object model: 0.1, 0.3, 0.5, 0.7, and 0.9. We let them use their own best threshold for each instrument because they are quite different models so they may have different optimal thresholds. We use 0.5 as the threshold in the sound model for SOT models.

	Temporal (AUC)				Spatial (Pixel)			
Action model	OT		SOT		OT		SOT	
Object model	Weakly supervised	Faster R-CNN	Weakly supervised	Faster R-CNN	Weakly supervised	Faster R-CNN	Weakly supervised	Faster R-CNN
Accordion action	**0.890**	0.835	**0.894**	0.859	24.0	**23.0**	**23.2**	26.1
Cello action	**0.932**	0.913	**0.937**	0.921	**23.1**	23.5	**24.6**	26.7
Drum action	0.831	**0.858**	0.854	**0.886**	**40.3**	45.3	**40.7**	42.8
Flute action	**0.756**	0.689	**0.785**	0.744	**26.4**	31.4	**26.2**	31.6
Guitar action	0.788	**0.819**	**0.864**	0.860	29.2	**25.7**	27.3	**27.1**
Piano action	0.722	**0.761**	0.778	**0.791**	37.2	**33.6**	**31.9**	42.1
Saxophone action	**0.751**	0.737	**0.754**	0.753	**23.2**	23.9	**25.0**	26.1
Trumpet action	**0.772**	0.751	0.777	**0.796**	**34.3**	38.1	35.5	**34.2**
Violin action	**0.891**	0.883	**0.904**	0.890	**17.8**	21.0	**17.7**	21.3
Average	**0.815**	0.805	**0.838**	0.833	**28.4**	29.5	**28.0**	30.9

proves the original action detection result using only the action model. For the temporal performance, the SOT without fusion achieves 0.834 AUC on average, while SOT$\times$Object$\times$Sound improves to 0.917 AUC. For the spatial performance, the SOT without fusion achieves 29.1 pixels, while the SOT$\times$Object improves to 25.3 pixels. The action detections of the instruments in the wind family are again among those that obtain the largest gains. We can also see that SOT still outperforms VT after fusion, which again verifies the importance of providing better supervisions. The temporal results are much closer after fusion for SOT and VT, probably because the sound model dominates the fused prediction. The spatial result of SOT still outperforms VT by a large margin.

4.3.5 Learned movements in the action model

We want to see what movements are learned in the action model. Specifically, we derive the stacked average directions from the learned filters in the first convolution layer, Conv1. The procedure is shown in Fig 4.9. A filter of Conv1 in an action model is a $10 \times 7 \times 7$ tensor for processing the stacked dense optical flows of five consecutive frames,

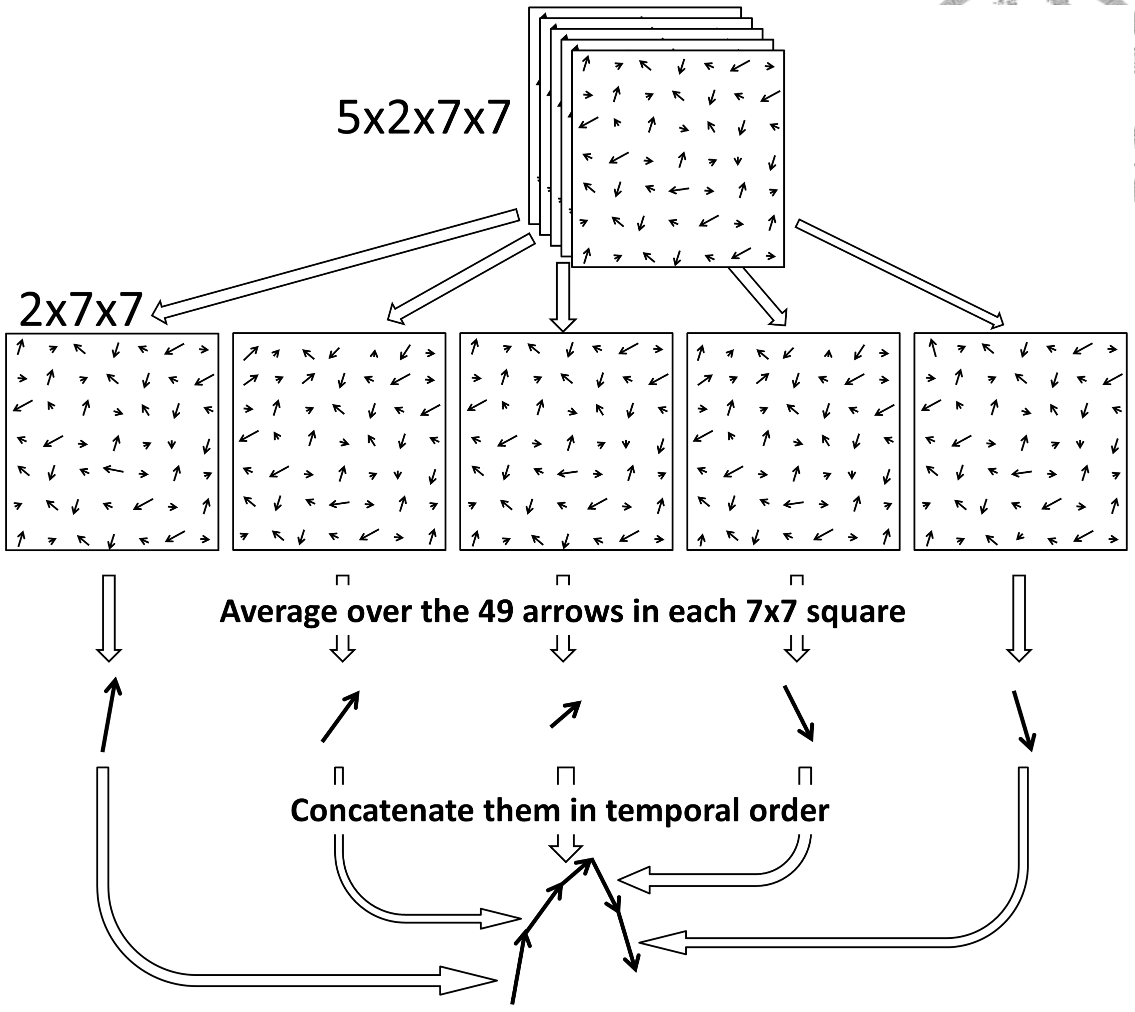

5x2x7x7
2x7x7
Average over the 49 arrows in each 7x7 square
Concatenate them in temporal order

Figure 4.9: Procedure of deriving stacked average movements.

which can also be seen as five $2 \times 7 \times 7$ tensors. By averaging over the 7×7 receptive field for each of the five tensors, we get five averaged movements. We derive the stacked average movements by concatenating these five averaged movements. This is done for each of the 96 filters in Conv1. The result is shown in Figure 4.10.

Next, we want to see what are the characterizing stacked average movements for the nine instruments, that is, the movements that occur relatively more often in one instrument than in other instruments. The procedure is shown in Figure 4.11. We collect the positive frames in the manually-annotated test data of the nine instruments. For the m-th instrument and the n-th filter of Conv1, we compute the spatial maximum of the rectified output for each positive frame, and take average over all these max values. This is done for all (m, n) pairs, and form a 9×96 matrix. Finally, we z-score normalize the matrix along the instrument dimension and sort along the filter dimension to get the top filters.

doi:10.6342/NTU201801365

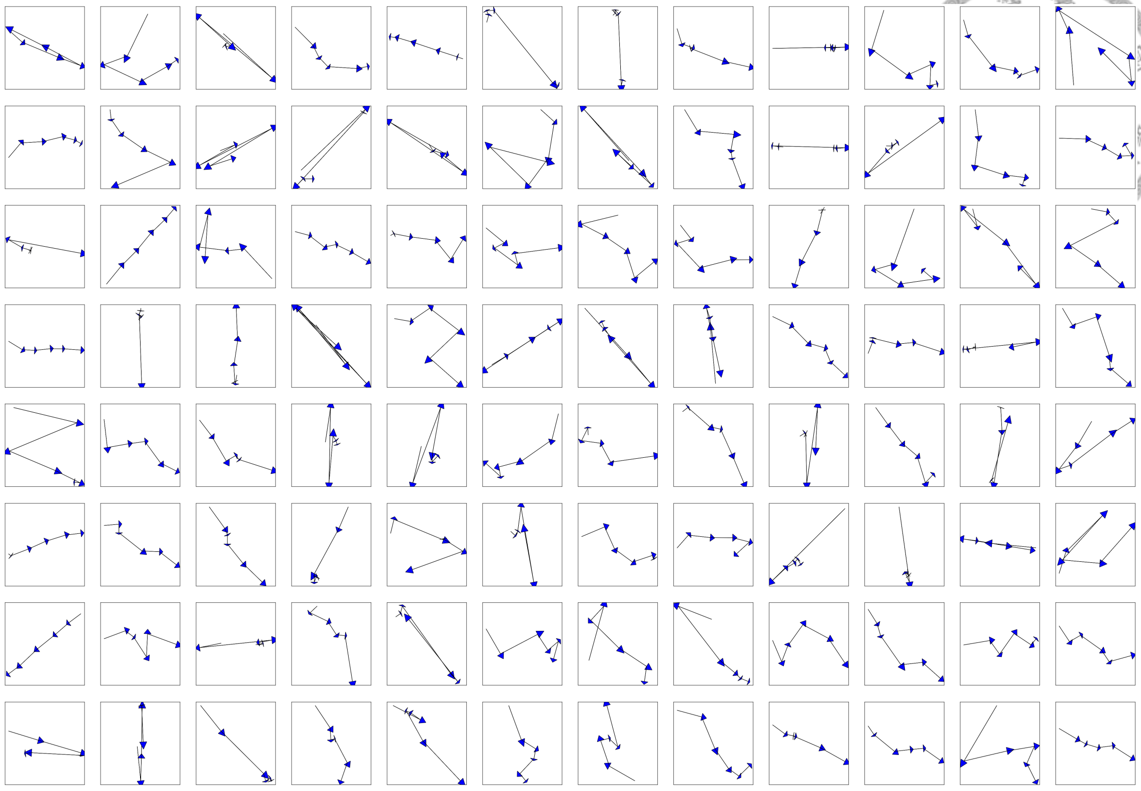


Figure 4.10: Stacked average movements derived from 96 filters of Conv1 (action model SOT0503).

The top five filters after this procedure are shown in Figure 4.12.

We can see several interesting patterns in this visualization. The characterizing movements of Accordion are all horizontal. Cello has mostly smooth movements but also contains some twisting in some movements. Drum and Guitar have more back-and-forth patterns, indicating that they have faster movements than the others. Most of the movements in Violin are smooth, similar to those in Cello.

4.3.6 Analyses and observations

In this subsection, I share some analyses and observations I have made in the process of doing experiments.

From Table 4.9 and Table 4.10, we can see that the Drum action has moderate temporal performance but the worst spatial performance among the nine instruments. In other words, the model can usually correctly detect the timing of drum playing, but predict at the wrong spatial location. This result is sensible because the contact of the stick and the

doi:10.6342/NTU201801365

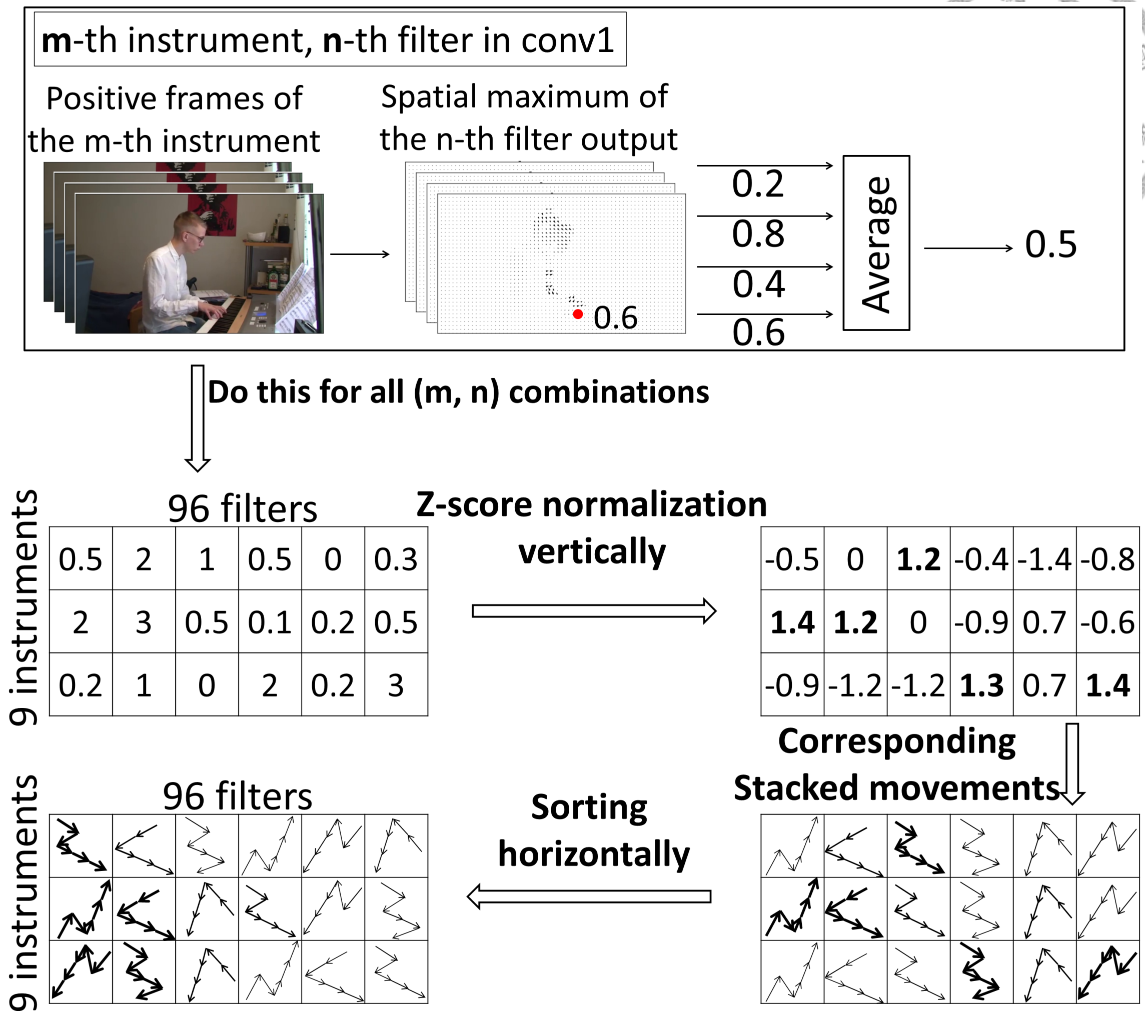


Figure 4.11: Procedure of deriving the characterizing stacked average movements for the instruments.

drum skin usually lasts for a very short period of time. After the contact, the stick would go somewhere else rapidly. Therefore, it is too fast for the model to correctly locate the contact point.

It is relatively more difficult to detect the playing timing of instruments in the wind family, such as Flute, Saxophone, and Trumpet, because the playing actions of these instruments are usually subtle finger movements. This is also reflected by the temporal performance shown in Table 4.9. Flute has another disadvantage due to its thin body. The object model often cannot well detect the Flute object, and this in turn affects the performance of the action models trained with the object model. In fact, Piano is another instrument that sometimes shares this difficulty because piano playing actions sometimes contain only finger movements while the hands do not move. I believe this is one of the

doi:10.6342/NTU201801365

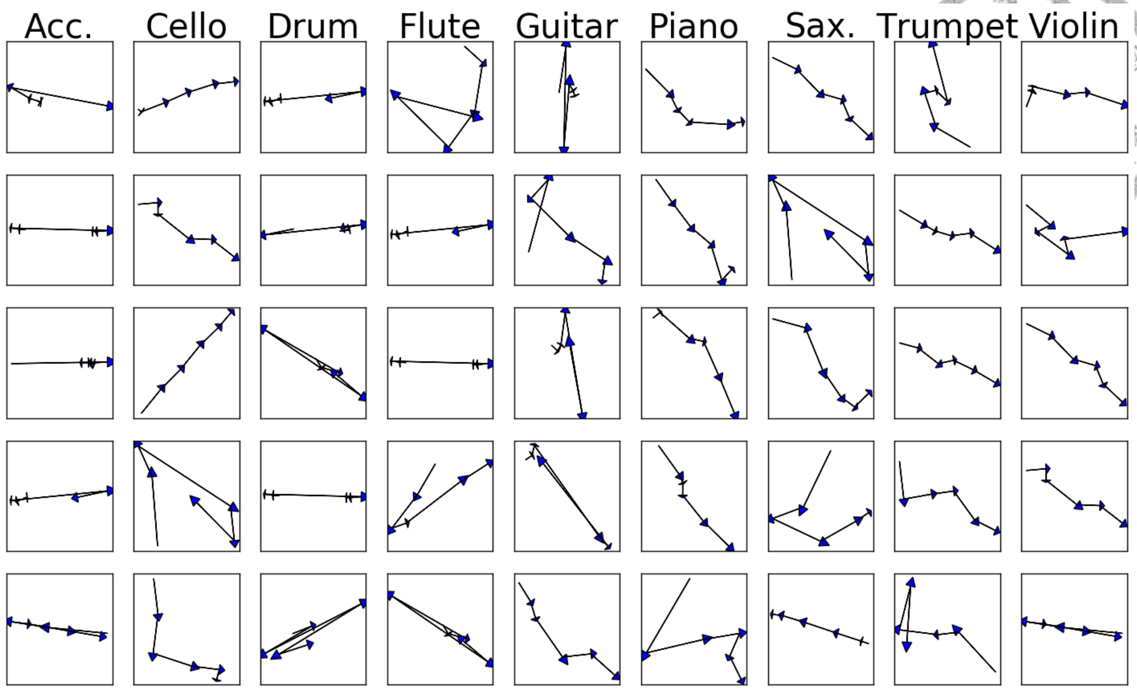


Figure 4.12: Characterizing stacked average movements of the 9 instruments.

reasons that the model does not perform well temporally on Piano.

Another interesting detection error of piano playing is from the optical reflection of hands on the polished cover of the pianos. We observe in several videos that the hands are reflected on the shiny black cover and the movements of the hands are also reflected on the cover. Both the object model and the action model can make false positives due to the optical reflection.

Originally, I thought that the playing actions of violin and cello might be similar and the model might have difficulty distinguishing them. Surprisingly, they are usually detected correctly without confusion.

I have seen ego-motions of cameras in many instrument-playing videos, and their effect to the action model would depend on several factors. First of all, I found that the music-related videos with ego-motions are usually filmed under worse conditions with poor resolutions because they are made by less professional people and in less formal occasions. Therefore, the problem of ego-motions is often coupled with worse video resolutions and serious occlusions. When these ill conditions come together, the model usually fails.

Although we do not consider the ego-motion when conducting the experiments, the

doi:10.6342/NTU201801365

model could handle the ego-motion in some circumstances. Assume we are in a scene where the player is not playing the instrument and there are ego-motions. We can handle the ego-motions in this situation if we use the fused prediction that combines the information of the sound activation into the action prediction, such as Action×Sound or Action×Object×Sound discussed in Section 4.1.3, because the action activations caused by ego-motions will be suppressed by the low activations of the sound model.

4.4 Summary

In this paper, a weakly-supervised framework is proposed to train a model for detecting instrument-playing actions in videos. In this framework, an auxiliary sound model and an auxiliary object model are utilized to provide supervisions to alleviate the lack of annotated data. It has been shown that the proposed framework can significantly improve the detection ability both temporally and spatially.

There are several possible future directions. First, subtle finger movements are important cues for playing actions of many instruments, and the current model might not be able to fully capture them. One possible way to improve it is to automatically detect hand gestures or mouth movements and use them as input features. Another possible way is to incorporate pose information into the framework. For example, packages like OpenPose [13] can estimate poses of bodies, fingers, and mouths. We may utilize the movements and locations of fingers and mouths to detect subtle playing actions. Second, the proposed framework could also be applied to other categories of actions where sounds, objects, or both are important cues of the actions.

[13] https://github.com/CMU-Perceptual-Computing-Lab/openpose

doi:10.6342/NTU201801365

Chapter 5

Conclusions and discussions

In previous chapters, we have seen how weakly-supervised event detection in music audios can be done with the help of FCNs, and we have also shown how event detection in music audios can help weakly-supervised action detection in msuic videos. Some possible future works are also discussed.

Although we mainly discuss the weakly-supervised approaches that are applied to music-related audios and videos, the proposed methods could be used in other problems as well. For example, we have seen in Section 3.5 that the model for music audios can also be used on daily-life audio events. The approach proposed for action detection in instrument playing also has the potential to be used in actions that involve sounds, such as violent actions.

The interactions between different modalities in videos provide many opportunities for learning tasks. We may think of videos as a simplified records for our daily lives. They include two of our most important perceptual inputs: audio information and visual information. The temporal dimension of videos adds another dimension of richness. Recently, we can see more and more studies working on various learning problems that utilize weakly-supervised or unsupervised approaches in videos. For example, there are unsupervised feature learning with videos and unsupervised audio source separation with videos. The instrument-playing action detection presented in this work is also among them. I believe there are many more topics that can be approached by exploiting the multi-modalities of videos.

doi:10.6342/NTU201801365

Appendix A

A GUI for displaying the result of music audio event detection

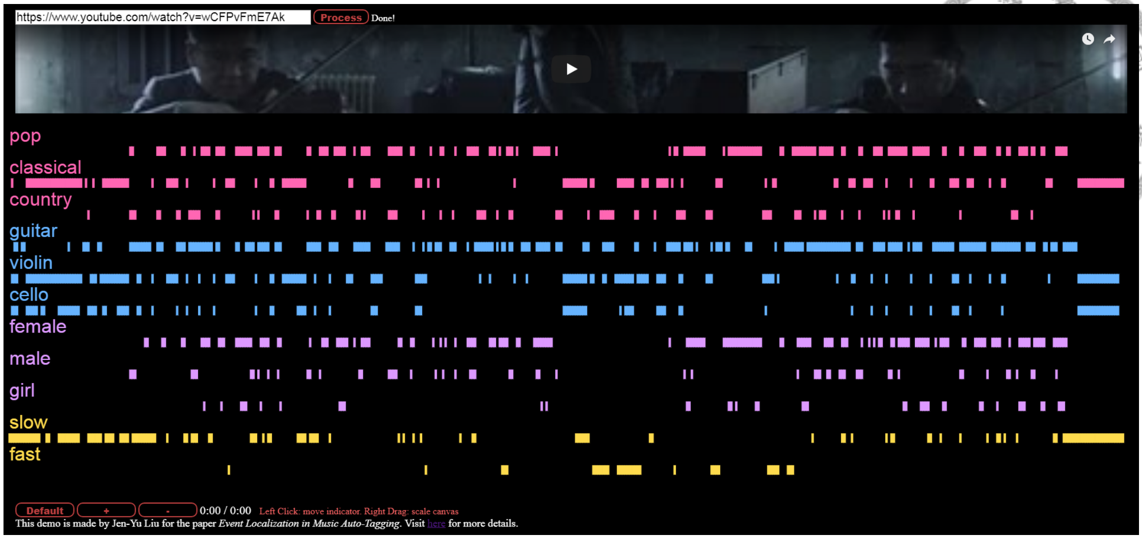
A web-based graphical user interface is built for demonstration. Link: http://clip2frame.ciaua.com. It is constructed using HTML, Javascript, and P5.js[1]. It takes an YouTube link, and displays the predictions of the proposed model. A sample screen shot is shown in Figure A.1. The 188 labels are grouped into four types: Genre, Instrument, Vocal, and Tempo. In each type, the top labels are displayed in a descending order. With this GUI, we can subjectively evaluate the models. Several observations have been made.

First, vocals can be recognized in general, but often have confusions between female vocal and male vocal. Figure A.2 is the prediction of a song with both female and males singers. The predictions can in general detect male and female voices correctly, even the speaking parts. The female and male vocals are more "typical" in the sense that the female singer has relatively higher pitches while the male singer has relatively lower pitches. When a male singer is in higher pitches, the model is often confused. For example, Figure A.3 shows such a case from Queen. Its vocalist, Freddie, could sing with high pitches. We can see that the model almost always recognizes his voices as female.

Guitar and piano sounds are often confused. For example, Figure A.4 shows the de-

[1]https://p5js.org/
[2]劉若英 - 為愛痴狂, https://www.youtube.com/watch?v=ZfTRmDgJZJA

doi:10.6342/NTU201801365



Left Click: move indicator. Right Drag: scale canvas

This demo is made by Jen-Yu Liu for the paper *Event Localization in Music Auto-Tagging*. Visit here for more details.

Figure A.1: A GUI for demonstrating the proposed model

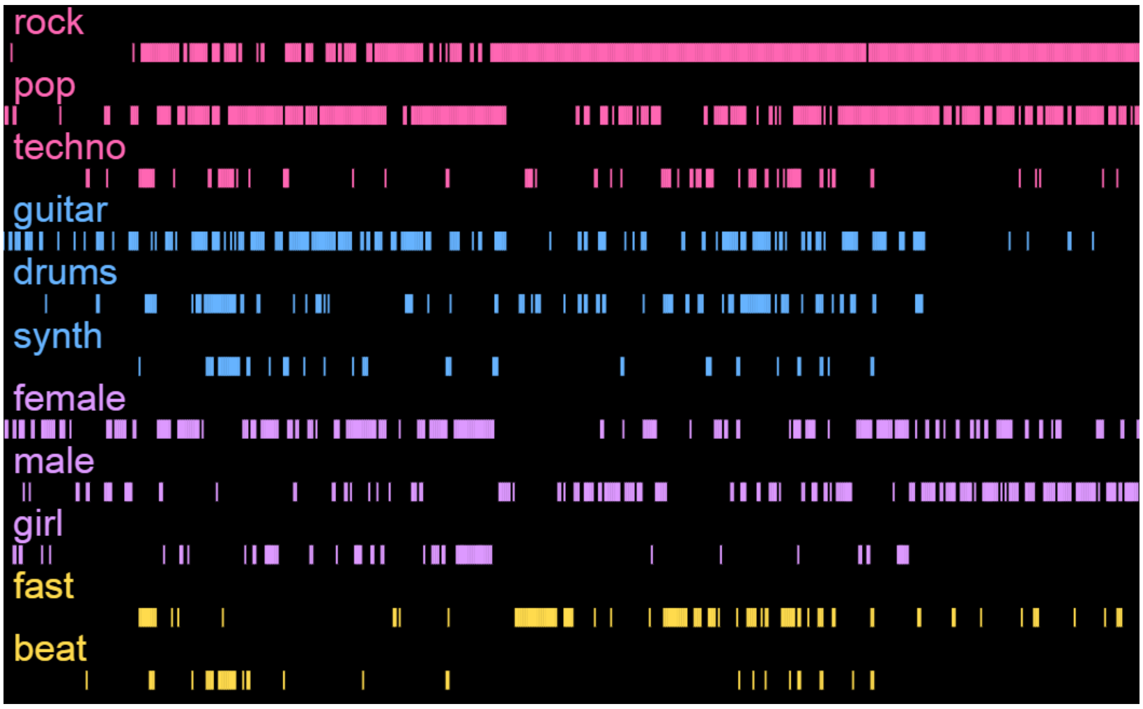


Figure A.2: A song with female and male vocals [2]

tection result of Tears in Heaven by Eric Clapton. Eric Clapton uses only guitar in this song, but the prediction detects several piano sounds. Subjectively, I have found in many songs that electrical guitar sounds could sound similar to piano sounds when they are recorded with high-quality guitar and high-quality recording environments, so sometimes it is even difficult for human beings to distinguish them. The other possible cause of the mis-detection is the design of the model. FCNs are used in this work. The convolution operation in FCNs can only see a small portion of the music audio (around 0.5 second in

doi:10.6342/NTU201801365

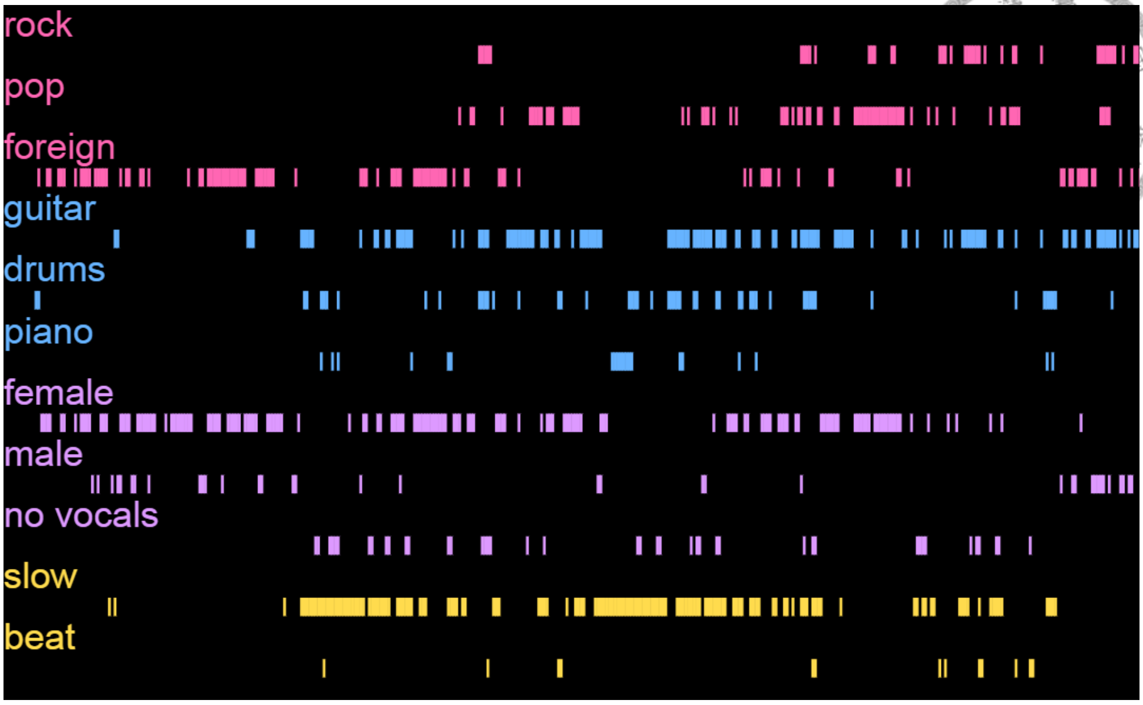


Figure A.3: Confusion between high-pitched male voices and female voices[3]

this work). This could result in the lack of key information for distinguishing guitar from piano. The sounds of the attack, such as the sound of a pick hitting the string, is important for us to distinguish a guitar sound from a piano sound, but this information could be missing when the model only sees a small part of the audio.

It is also observed that guitar and male vocal are sometimes confused. Figure A.5 shows such a case. This song from Def Tech has only male vocal before the red line, but the model detects guitar sounds. My conjecture is that this is a result of the characteristic of the training dataset MagnaTagATune and the use of weakly-supervised learning. MagnaTagATune mostly consists of Pop songs and Rock songs, and has relatively few songs of pure instrumental guitar. This leads to the result that guitar sounds are usually accompanied with singing, especially male singing. Therefore, the label 'guitar' and 'male' often appear together in the same clip or song. Furthermore, we have only clip-level annotation in the weakly-supervised setting. This could make it difficult for the model to distinguish between 'guitar' and 'male.'

Violin and cello consist of another pair that is often confused. An example of cello

[3]Queen - Bohemian Rhapsody, https://www.youtube.com/watch?v=fJ9rUzIMcZQ
[4]Eric Clapton - Tears in Heaven, https://www.youtube.com/watch?v=VmLQes4tmtM
[5]Def Tech - My Way, https://www.youtube.com/watch?v=AbHzv3QeEC0

doi:10.6342/NTU201801365

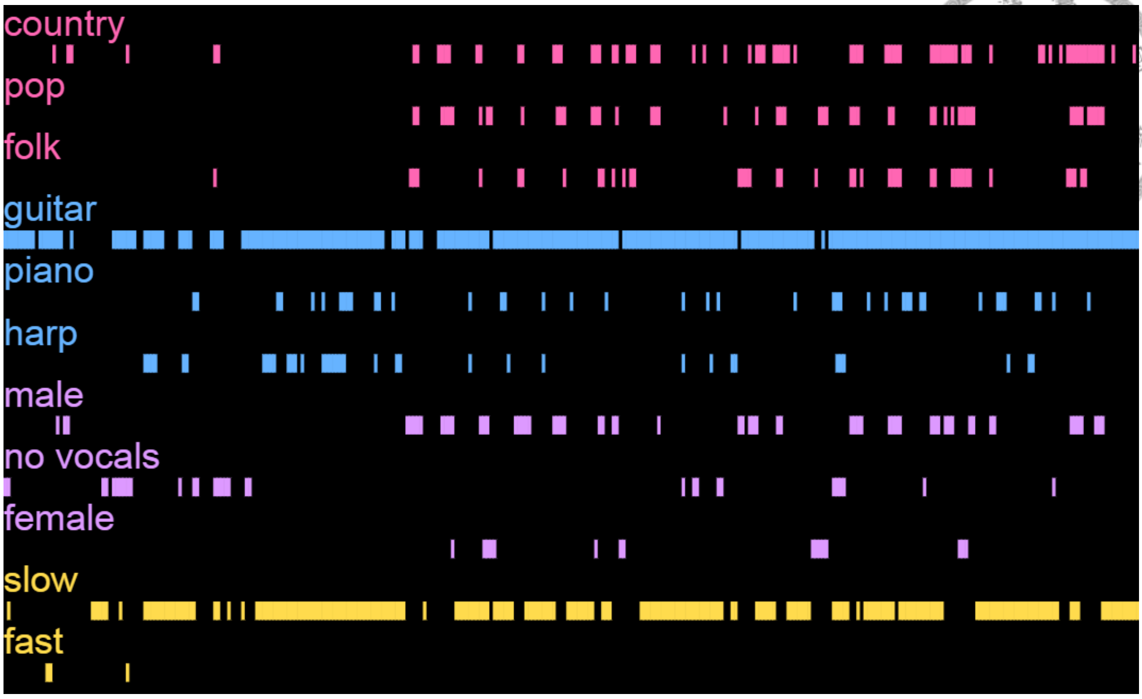


Figure A.4: Confusion between guitar and piano[4]

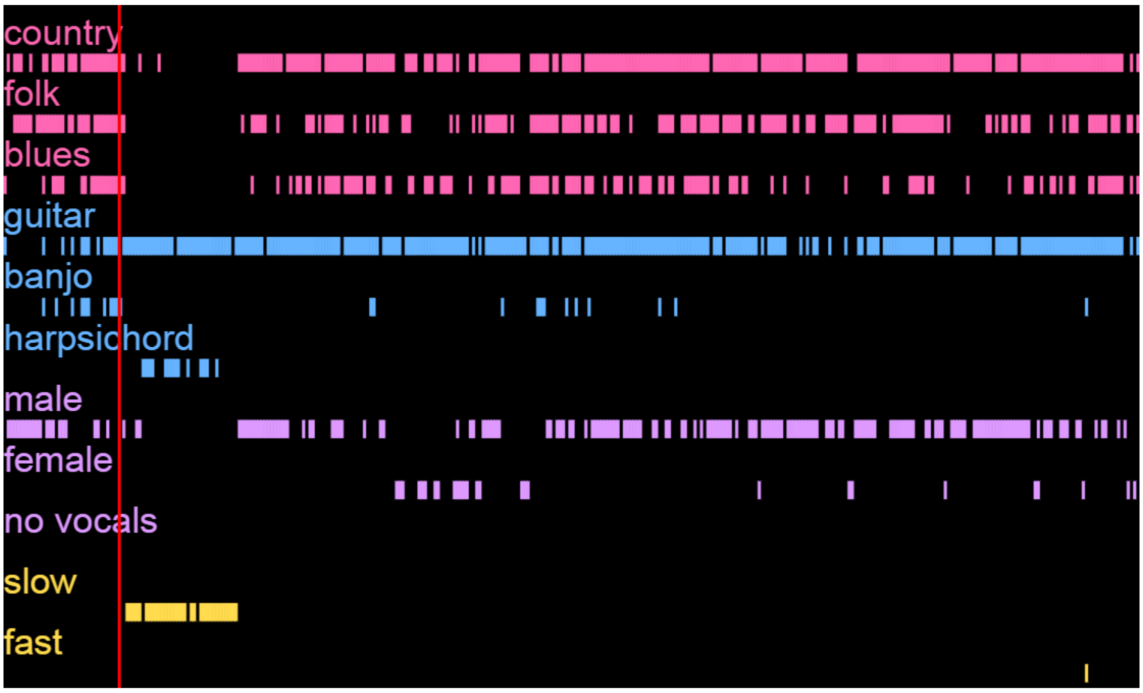


Figure A.5: Confusion between guitar and male voices[5]

playing is shown in Figure A.6, and most parts of the cello playing are also detected as violin playing by the model. This confusion is in fact understandable because the higher-pitched part of a cello and the lower-pitched part of a violin can sound similar. I have found that I am able to distinguish between violin and cello sounds in many cases because

doi:10.6342/NTU201801365

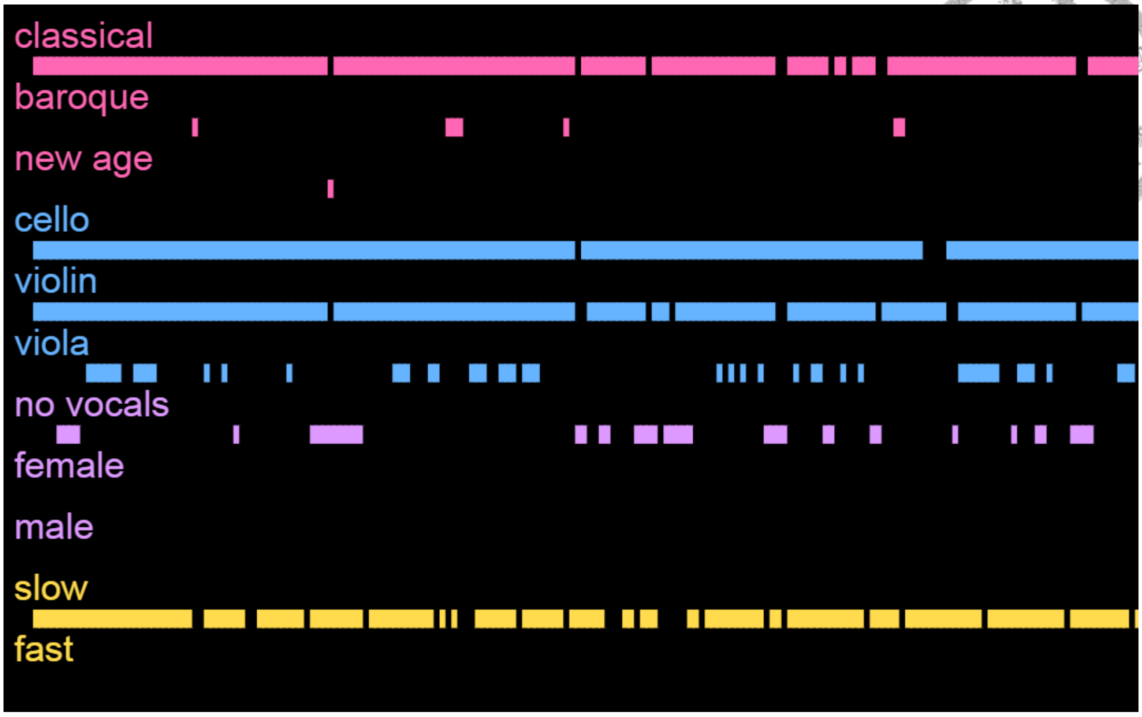


Figure A.6: Confusion between violin and cello[6]

I implicitly maintain the assumption that the instruments in a period of time are basically fixed. Therefore, I can take into account the range of notes I hear in a period of time and use this information to make a judgment about whether it is a violin or a cello. This indicates that we human beings would take into account some global information about a song when we try to recognize the instruments. The proposed models do not have such an ability, and this can be a direction for future work.

Predictions on Genre are usually good. One reason is that genre is a more subjective property.

[6]Ophélie Gaillard, Bach, Prelude, Cello suite n. 1, https://www.youtube.com/watch?v=poCw2CCrfzA

doi:10.6342/NTU201801365

Appendix B

Publications

1. **Liu, J.-Y.**, Yang, Y.-H., and Jeng, S.-K. (2018). Weakly-supervised Visual Instrument-playing Action Detection in Videos. IEEE Transactions on Multimedia (AQ: Accepted with Minor Revision).

2. Su, T.-W., **Liu, J.-Y.**, and Yang, Y.-H. (2017). Weakly-supervised Audio Event Detection Using Event-specific Gaussian Filters and Fully-Convolutional Networks. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

3. Yang, L.-C., Chou, S.-Y., **Liu, J.-Y.**, Yang, Y.-H., and Chen, Y.-A. (2017). Revisiting the problem of audio-based hit song prediction using convolutional neural networks. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 621–625). IEEE.

4. **Liu, J.-Y.**, and Yang, Y.-H. (2016). Event Localization in Music Auto-tagging. In ACM international conference on Multimedia (MM).

5. **Liu, J.-Y.**, Jeng, S.-K., and Yang, Y.-H. (2016). Applying Topological Persistence in Convolutional Neural Network for Music Audio Signals. arXiv Preprint.

6. Su, L., Yeh, C.-C., **Liu, J.-Y.**, Wang, J.-C., and Yang, Y.-H. (2014). A systematic evaluation of the bag-of-frames representation for music information retrieval. IEEE Transactions on Multimedia, 16(5), 1188–1200.

doi:10.6342/NTU201801365

7. **Liu, J.-Y.**, Liu, S.-Y., and Yang, Y.-H. (2014). LJ2M dataset: Toward better understanding of music listening behavior and user mood. In IEEE International Conference on Multimedia and Expo (ICME) (pp. 1–6). IEEE.

8. Yang, Y.-H., and **Liu, J.-Y.** (2013). Quantitative Study of Music Listening Behavior in a Social and Affective Context. IEEE Transactions on Multimedia, 15(6), 1304–1315.

9. **Liu, J.-Y.**, and Yang, Y.-H. (2012). Inferring personal traits from music listening history. In ACM workshop on Music information retrieval with user-centered and multimodal strategies (MIRUM) (p. 31). New York, New York, USA: ACM Press.

10. **Liu, J.-Y.**, Yeh, C.-C., Yang, Y.-H., and Teng, Y.-C. (2012). Bilingual analysis of song lyrics and audio words. In ACM international conference on Multimedia (MM) (p. 829). New York, New York, USA: ACM Press.

doi:10.6342/NTU201801365

Bibliography

[1] Jen-Yu Liu and Yi-Hsuan Yang. Event localization in music auto-tagging. In *Proceedings of the ACM international conference on Multimedia (MM)*, 2016.

[2] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: delving deep into convolutional nets. *British Machine Vision Conference*, pages 1–11, may 2014.

[3] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6): 82–97, 2012.

[4] Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer, 2012.

[5] Ossama Abdel-Hamid, Li Deng, and Dong Yu. Exploring convolutional neural network structures and optimization techniques for speech recognition. In *Proceedings of INTERSPEECH*, pages 3366–3370, 2013.

[6] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545, Oct 2014.

doi:10.6342/NTU201801365

[7] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.

[9] Pedro O Pinheiro and Ronan Collobert. From image-level to pixel-level labeling with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1713–1721, 2015.

[10] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free? - Weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 685–694, 2015.

[11] George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1742–1750, 2015.

[12] Philippe Hamel, Simon Lemieux, Yoshua Bengio, and Douglas Eck. Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 729–734, 2011.

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

doi:10.6342/NTU201801365

[14] Sander Dieleman and Benjamin Schrauwen. Multiscale approaches to music audio feature learning. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 116–121, 2013.

[15] Douglas Turnbull Derek Tingle, Youngmoo E. Kim. Exploring automatic music nnotation with acoustically-objective tags. In *Proceedings of the ACM International Conference on Multimedia Retrieval (ICMR)*, pages 55–61, 2010.

[16] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett. Darpa timit acoustic phonetic continuous speech corpus. 1993. *US Dept. of Commerce, NIST, Gaithersburg, USA*.

[17] Edith Law, Kris West, Michael I. Mandel, Mert Bay, and J. Stephen Downie. Evaluation of algorithms using games: the case of music tagging. In *Proceedings of the International Society of Music Information Retrieval Conference (ISMIR)*, 2009.

[18] Rachel M. Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. Medleydb: A multitrack dataset for annotation-intensive mir research. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 155–160, 2014. http://medleydb.weebly.com.

[19] Zhong Zhou, Feng Shi, and Wei Wu. Learning spatial and temporal extents of human actions for action detection. *IEEE Transactions on Multimedia (TMM)*, 17(4):512–525, apr 2015.

[20] Yemin Shi, Yonghong Tian, Yaowei Wang, and Tiejun Huang. Sequential deep trajectory descriptor for action recognition with three-stream CNN. *IEEE Transactions on Multimedia (TMM)*, 19(7):1510–1520, jul 2017.

[21] Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Weakly supervised action labeling in videos under ordering constraints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 8693 LNCS, pages 628–643. Springer, Cham, 2014.

doi:10.6342/NTU201801365

[22] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1–11, 2014.

[23] Joe Yue Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 07-12-June, pages 4694–4702, 2015.

[24] Yu-Gang Jiang, Qi Dai, Tao Mei, Yong Rui, and Shih-Fu Chang. Super fast event recognition in internet videos. *IEEE Transactions on Multimedia (TMM)*, 17(8): 1174–1186, aug 2015.

[25] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, 2016.

[26] De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Connectionist temporal modeling for weakly supervised action labeling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 137–153. Springer International Publishing, 2016.

[27] Bochen Li, Karthik Dinesh, Zhiyao Duan, and Gaurav Sharma. See and listen: Score-informed association of sound tracks to players in chamber music performance videos. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2906–2910. IEEE, mar 2017.

[28] Weiming Hu, Xinmiao Ding, Bing Li, Jianchao Wang, Yan Gao, Fangshi Wang, and Stephen Maybank. Multi-perspective cost-sensitive context-aware multi-instance sparse coding and its application to sensitive video recognition. *IEEE Transactions on Multimedia (TMM)*, 18(1):76–89, jan 2016.

doi:10.6342/NTU201801365

[29] Francesco Cricri, Mikko J. Roininen, Jussi Leppanen, Sujeet Mate, Igor D. D. Curcio, Stefan Uhlmann, and Moncef Gabbouj. Sport type classification of mobile videos. *IEEE Transactions on Multimedia (TMM)*, 16(4):917–932, jun 2014.

[30] Xu Cheng, Jiangchuan Liu, and Cameron Dale. Understanding the characteristics of internet short video sharing: A YouTube-based measurement study. *IEEE Transactions on Multimedia (TMM)*, 15(5):1184–1194, aug 2013.

[31] Slim Essid, Gaël Richard, and Bertrand David. Instrument recognition in polyphonic music based on automatic taxonomies. *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, 14(1):68–80, jan 2006.

[32] Zhouyu Fu, Guojun Lu, Kai Ming Ting, and Dengsheng Zhang. A survey of audio-based music classification and annotation. *IEEE Transactions on Multimedia (TMM)*, 13(2):303–319, 2011.

[33] Dimitrios Giannoulis, Emmanouil Benetos, Anssi Klapuri, and Mark D. Plumbley. Improving instrument recognition in polyphonic music through system integration. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5222–5226. IEEE, may 2014.

[34] Yoonchang Han, Jaehun Kim, and Kyogu Lee. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, 25(1):208–221, jan 2017.

[35] Antonello Rizzi, Mario Antonelli, and Massimiliano Luzi. Instrument learning and sparse NMD for automatic polyphonic music transcription. *IEEE Transactions on Multimedia (TMM)*, 19(7):1405–1415, jul 2017.

[36] Loïc Reboursière, Otso Lähdeoja, Thomas Drugman, Stéphane Dupont, Cecile Picard-Limpens, and Nicolas Riche. Left and right-hand guitar playing techniques detection. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2012.

doi:10.6342/NTU201801365

[37] Alexander Schindler and Andreas Rauber. Harnessing music-related visual stereotypes for music information retrieval. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):1–21, oct 2016.

[38] Olga Slizovskaia, Emilia Gómez, and Gloria Haro. Musical instrument recognition in user-generated videos using a multimodal convolutional neural network architecture. In *Proceedings of the ACM International Conference on Multimedia Retrieval (ICMR)*, pages 226–232, New York, New York, USA, 2017. ACM Press.

[39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[40] Soobeom Jang and Jong-Seok Lee. On evaluating perceptual quality of online user-generated videos. *IEEE Transactions on Multimedia (TMM)*, 18(9):1808–1818, sep 2016.

[41] Douglas Eck, Paul Lamere, Stephen Green, and Thierry Bertin-Mahieux. Automatic generation of social tags for music recommendation. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1–8, 2007.

[42] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):467–476, feb 2008.

[43] Mikael Henaff, Kevin Jarrett, Koray Kavukcuoglu, and Yann LeCun. Unsupervised learning of sparse features for scalable audio classification. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 681–686, 2011.

[44] Juhan Nam, Jorge Herrera, Malcolm Slaney, and Julius O. Smith. Learning sparse feature representations for music annotation and retrieval. In *Proceedings of the*

doi:10.6342/NTU201801365

International Society of Music Information Retrieval Conference (ISMIR), pages 565–570, 2012.

[45] Eric J. Humphrey, Juan Pablo Bello, and Yann LeCun. Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In *Proceedings of the International Society of Music Information Retrieval Conference (ISMIR)*, pages 403–408, 2012.

[46] Slim Essid, Gaël Richard, and Bertrand David. Instrument recognition in polyphonic music based on automatic taxonomies. *IEEE Transactions on Audio, Speech and Language Processing*, 14(1):68–80, jan 2006.

[47] Michael I. Mandel and Daniel P.W. Ellis. A Web-Based Game for Collecting Music Metadata. *Journal of New Music Research*, 37(2):151–165, 2008.

[48] Michael I. Mandel, Douglas Eck, and Yoshua Bengio. Learning tags that vary within a song. pages 399–404, 2010.

[49] Michael I. Mandel, Razvan Pascanu, Douglas Eck, Yoshua Bengio, Luca M. Aiello, Rossano Schifanella, and Filippo Menczer. Contextual tag inference. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 7S(1):1–18, oct 2011.

[50] Shuo-Yang Wang, Ju-Chiang Wang Yi-Hsuan Yang, and Hsin-Min Wang. Towards time-varying music auto-tagging based on cal500 expansion. In *Proceedings of the IEEE International Conference on Multimedia and Expo. (ICME)*, pages 1–6, 2014.

[51] Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. Recurrent neural networks for polyphonic sound event detection in real life recordings. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

doi:10.6342/NTU201801365

[52] Anurag Kumar and Bhiksha Raj. Audio event detection using weakly labeled data. In *Proceedings of the ACM international conference on Multimedia (MM)*, pages 1038–1047. ACM Press, 2016.

[53] Jan Schlüter. Learning to pinpoint singing voice from weakly labeled examples. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 44–50, 2016.

[54] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[55] Alex Krizhevsky, Ilya Sutskever, and Hinton Geoffrey E. ImageNet classification with deep convolutional neural networks. *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1–9, 2012.

[56] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[57] Glenn Hartmann, Matthias Grundmann, Judy Hoffman, David Tsai, Vivek Kwatra, Omid Madani, Sudheendra Vijayanarasimhan, Irfan Essa, James Rehg, and Rahul Sukthankar. Weakly supervised learning of object segmentations from web-scale video. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 7583 LNCS, pages 198–208. Springer, Berlin, Heidelberg, 2012.

[58] Xiao Liu, Dacheng Tao, Mingli Song, Ying Ruan, Chun Chen, and Jiajun Bu. Weakly supervised multiclass video segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 57–64, 2014.

[59] Alessandro Prest, Christian Leistner, Javier Civera, Cordelia Schmid, and Vittorio Ferrari. Learning object class detectors from weakly annotated video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3282–3289. IEEE, jun 2012.

doi:10.6342/NTU201801365

[60] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. Cross modal distillation for supervision transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2827–2836, 2016.

[61] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. SoundNet: Learning sound representations from unlabeled video. *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, oct 2016.

[62] Alfredo Canziani and Eugenio Culurciello. CortexNet: A generic network family for robust visual temporal representations. *arXiv preprint*, jun 2017.

[63] Relja Arandjelović and Andrew Zisserman. Look, listen and learn. *arXiv preprint*, may 2017.

[64] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. See, hear, and read: Deep aligned representations. *arXiv preprint*, jun 2017.

[65] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732. IEEE, jun 2014.

[66] Zhongwen Xu, Yi Yang, and Alex G. Hauptmann. A discriminative cnn video representation for event detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[67] Zhi-Hua Zhou and Min-Ling Zhang. Multi-instance multi-label learning with application to scene classification. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1609–1616. MIT Press, 2007.

[68] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. Semantic segmentation with boundary neural fields. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

doi:10.6342/NTU201801365

[69] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, 10(5): 293–302, 2002.

[70] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, jan 2018.

[71] Serena Yeung, Vignesh Ramanathan, Olga Russakovsky, Liyue Shen, Greg Mori, and Li Fei-Fei. Learning to learn from noisy web videos. *arXiv preprint*, 2017.

[72] Anna Khoreva, Rodrigo Benenson, Jan Hosang, Matthias Hein, and Bernt Schiele. Simple does it: Weakly supervised instance and semantic segmentation. 2016.

[73] Jiajun Wu, Yu Yinan, Chang Huang, and Yu Kai. Deep multiple instance learning for image classification and auto-annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3460–3469. IEEE, 2015.

[74] Ehsan Adeli Mosabbeb, Ricardo Cabral, Fernando De la Torre, and Mahmood Fathy. Multi-label discriminative weakly-supervised human activity recognition and localization. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, volume 9007, pages 241–258. 2015.

[75] Ting-Wei Su, Jen-Yu Liu, and Yi-Hsuan Yang. Weakly-supervised audio event detection using event-specific Gaussian filters and fully-convolutional networks. *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[76] Jurgen Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3642–3649, 2012.

[77] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 807–814, 2010.

doi:10.6342/NTU201801365

[78] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, pages 1–14, sep 2015.

[79] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[80] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In Josef Bigun and Tomas Gustavsson, editors, *Image Analysis*, pages 363–370. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[81] Deepak Pathak, Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional multi-class multiple instance learning. *arXiv preprint arXiv:1412.7144*, 2014.

[82] Jonathan Masci, Alessandro Giusti, Dan Ciresan, Gabriel Fricout, and Jürgen Schmidhuber. A fast learning algorithm for image segmentation with max-pooling convolutional networks. In *Proceedings on the IEEE International Conference on Image Processing (ICIP)*, pages 2713–2717, 2013.

[83] Pedro O Pinheiro and Ronan Collobert. Recurrent convolutional neural networks for scene labeling. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 82–90, 2014.

[84] Brian McFee, Matt McVicar, Colin Raffel, Dawen Liang, Oriol Nieto, Eric Battenberg, Josh Moore, Dan Ellis, Ryuichi Yamamoto, Rachel Bittner, Douglas Repetto, Petr Viktorin, João Felipe Santos, and Adrian Holovaty. librosa: 0.4.1, October 2015.

[85] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, and Desjardins. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. http://deeplearning.net/software/theano/.

doi:10.6342/NTU201801365

[86] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[87] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[88] Jia-Ching Wang, Chang-Hong Lin, Bo-Wei Chen, and Min-Kang Tsai. Gabor-based nonuniform scale-frequency map for environmental sound classification in home automation. *IEEE Transactions on Automation Science and Engineering*, 11(2): 607–613, April 2014.

[89] Pasquale Foggia, Nicolai Petkov, Alessia Saggese, Nicola Strisciuglio, and Mario Vento. Audio surveillance of roads: a system for detecting anomalous sounds. *IEEE Transactions on Intelligent Transportation Systems*, 17, 2015.

[90] Pierre Laffitte, David Sodoyer, Charles Tatkeu, and Laurent Girin. Deep neural networks for automatic detection of screams and shouted speech in subway trains. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6460–6464, March 2016.

[91] Tim Fischer, Johannes Schneider, and Wilhelm Stork. Classification of breath and snore sounds using audio data recorded with smartphones in the home environment. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 226–230, March 2016.

[92] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A dataset and taxonomy for urban sound research. In *Proceedings of the ACM International Conference on Multimedia (ACM MM)*, pages 1041–1044, New York, NY, USA, 2014. ACM.

doi:10.6342/NTU201801365

[93] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research (JMLR)*, 12:2121–2159, 2011.

[94] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[95] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

[96] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, mar 2017.

[97] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, apr 2015.

[98] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *Proceedings of Advances in Neural Information Processing Systems*, jun 2015.

[99] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common objects in context. may 2014.

[100] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[101] Brian McFee, Matt McVicar, Oriol Nieto, Stefan Balke, Carl Thome, Dawen Liang, Eric Battenberg, Josh Moore, Rachel Bittner, Ryuichi Yamamoto, Dan Ellis, Fabian-Robert Stoter, Douglas Repetto, Simon Waloschek, CJ Carr, Seth Kranzler, Keunwoo Choi, Petr Viktorin, Joao Felipe Santos, Adrian Holovaty, Waldir Pimenta, and Hojin Lee. librosa 0.5.0, February 2017.

doi:10.6342/NTU201801365

[102] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. YouTube-8M: A large-scale video classification benchmark. sep 2016.

[103] Jort F. Gemmeke, Daniel P. W. 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 *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, 2017.

[104] Zheng Shou, Dongang Wang, and Shih-Fu Chang. Temporal action localization in untrimmed videos via multi-stage CNNs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, jan 2016.

doi:10.6342/NTU201801365