Event Localization in Music Auto-tagging

Jen-Yu Liu[1,2], Yi-Hsuan Yang[2]
[1]Electrical Engineering Department, National Taiwan University, Taiwan
[2]Research Center for IT Innovation, Academia Sinica, Taiwan
{ciaua, yang}@citi.sinica.edu.tw

ABSTRACT

In music auto-tagging, people develop models to automatically label a music clip with attributes such as instruments, styles or acoustic properties. Many of these tags are actually descriptors of local events in a music clip, rather than a holistic description of the whole clip. Localizing such tags in time can potentially innovate the way people retrieve and interact with music, but little work has been done to date due to the scarcity of labeled data with granularity specific enough to the frame level. Most labeled data for training a learning-based model for music auto-tagging are in the clip level, providing no cues when and how long these attributes appear in a music clip. To bridge this gap, we propose in this paper a convolutional neural network (CNN) architecture that is able to make accurate frame-level predictions of tags in unseen music clips by using only clip-level annotations in the training phase. Our approach is motivated by recent advances in computer vision for localizing visual objects, but we propose new designs of the CNN architecture to account for the temporal information of music and the variable duration of such local tags in time. We report extensive experiments to gain insights into the problem of event localization in music, and validate through experiments the effectiveness of the proposed approach. In addition to quantitative evaluations, we also present qualitative analyses showing the model can indeed learn certain characteristics of music tags.

1. INTRODUCTION

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 [1,2,14,17]. 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 [11, 13, 28, 29, 33].

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 [7, 8, 15, 35]. 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 retrieval. For example, a user would not know 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, 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 [14, 17]. In visual object recognition, manual annotation of the so-called bounding boxes around the objects of interest is usually needed for training a model for localization [11]. 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 [12], it is typical to have multiple instruments, vocals, or styles in music, rendering it a multi-label problem [21]. As opposed to image, labeling music requires certain level of domain expertise. Localizing events in a time stream is also more labor-intense and time-consuming. In consequence, it is not surprising that most datasets available for music auto-tagging contain only clip-level annotations [21, 35].

In light of these observations, we propose in this paper a novel approach that conquers the scarcity of labeled data for music event localization by using only clip-level annotation of tags in the training phase. Our contribution is threefold. First, to the best of our knowledge, 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, we also present a series of experiments to gain insights into is-

MM '16, October 15-19, 2016, Amsterdam, Netherlands

DOI: http://dx.doi.org/10.1145/2964284.2964292

sues specific to the frame-level prediction task, namely music event localization. In general, "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 this paper, we achieve event localization by frame-level prediction, and we will use them interchangeably because we can derive the other if we have one.

Second, by extending a recent convolutional neural network (CNN)-based approach to visual object recognition [28], we develop a new CNN model for predicting and localizing 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 CNN. Different from existing CNN models for musical feature extraction [7,8], our model can deal with music of arbitrary length. The code is available online[1].

Third, as no evaluation framework has been proposed for music event localization trained with clip-level annotations, we are also motivated to propose such a framework to facilitate objective evaluation of the prediction result. We argue 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 [21] and a recently proposed multi-track dataset called MedleyDB [5]. We also present visualization of some prediction results, which can be used to subjectively evaluate the performance for both instrument and non-instrument tags.

The paper is organized as follows. We survey key related work in Section 2 and present the proposed method in Section 3. We then describe the datasets used in our evaluation in Section 4 and the evaluation result in Sections 5 and 6. Section 7 discusses and concludes the paper.

2. RELATED WORK

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) [7–9, 15, 16, 27, 35, 36]. 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 [7, 8, 18]. For example, Dieleman *et al.* have shown the effectiveness of CNN in learning features for music auto-tagging [8]. However, this CNN model can neither deal with music of arbitrary length nor perform frame-level prediction.

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 [10]. Mandel *et al.* study the tag relationships inside a track and between tags with 10-second clips [22–24]. Frame-level prediction on music auto-tagging was discussed by Wang *et al.* [37]. Parascandolo *et al.* conducts polyphonic sound event detection with recurrent neural networks [30]. The main difference between our methods and those in these studies is that our method can predict at a temporal granularity finer than the granularity of the training data.

Labeling the bounding boxes of objects in an image requires more labors than annotating the presence of objects does. Therefore, visual object localization with only image-level annotations has attracted increasing attentions in recent years [13, 28, 29, 33].

These approaches are sometimes referred to as *weakly supervised learning* [29]. Among the prior arts, our approach is closest to the model proposed by Oquab *et al.* [28], 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* (see Section 3 for details), 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, 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 [19,34,38]. 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.

3. MODEL

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 [2], computer vision [20], and also music information retrieval [8]. 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 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, where all the layers are convolutional [28]. 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 [28, 31], 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.

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

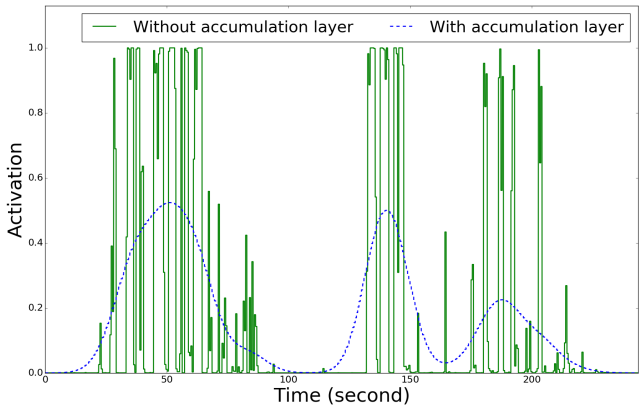


Figure 1: Expected effect of the accumulation layer.

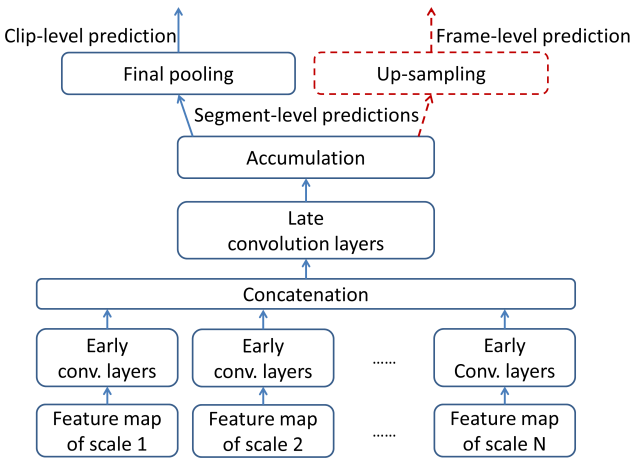


Figure 2: The proposed CNN architecture. Given a trained network for clip-level prediction (the blue one), we replace the final pooling layer by an up-sampling layer for frame-level prediction (the red component).

To account for these considerations, we propose to add an *accumulation* 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 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 paper, we realize the accumulation layer 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. We expect 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. We therefore leave it 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.* [7] 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 2. According to our design, the proposed model can make both clip-level and frame-level predictions. We provide the details in what follows.

3.1 Clip-level Prediction

As Figure 2 shows, the input layer consists of log mel-spectrogram with N different STFT window sizes [7]. 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 *stride size*. All stacks of early convolution layers in our CNN 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.

To facilitate our discussion later, 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.

3.2 Frame-level model

Given a trained network for clip-level prediction, we modify only the last layer to make it possible to make frame-level predictions, as illustrated also in Figure 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. We investigate two methods of up-sampling 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 [25,32,33]. 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 CNN network. This might be more accurate than the upscaling method, but obviously it is computationally much more expensive.

3.2.1 Thresholding

An important issue not depicted in Figure 2 is the necessity to apply a thresholding on the frame-level predictions to get binary decisions. This is because the output of CNNs 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, we empirically tune the values 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.

4. DATA

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

MagnaTagATune is a music dataset containing clip-level tag annotations [21]. The clips in the dataset were annotated by collecting, through playing a game, human evaluations of music tags that are generated by algorithm. It includes 25,863 29-second clips, annotated with 188 tags. The tags includes 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 [5]. It has instrument and F0 annotations at 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 paper, the multiple tracks of a clip are merged into one channel.

As there are no music auto-tagging dataset that contains frame-level annotations, we 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, we propose to use MedleyDB for evaluating frame-level result.

To ensure that the data are reasonably rich for both the training phase with MagnaTagATune and the evaluation phase with MedleyDB, we choose the top 9 common instrument tags 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."

4.1 Data processing

The audio feature used in our model is 128-D log mel-spectrogram, which has been used by [7, 8, 15]. All features are standardized by removing mean and divided by standard deviation derived from training set. We extract features with a Python library Librosa [26] with 16k sampling rate and hop size 512.

In processing MagnaTagATune, we follow [7, 8] 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 5 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.

5. EVALUATION

The models are implemented with Lasagne[2] and Theano [3, 4]. 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 from the 188-dimensional output. The CNNs 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 follows [8] 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 $TotalStride \times \sigma = 256$ (in 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 on MedleyDB, we compare the performance using thresholds derived from MagnaTagATune or from MedleyDB in the way described in Section 3.2.1. 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.

Due to the space limit, some details of the model implementation are omitted. Please see the documentation of our codes for the details.

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

Table 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

We will first present the best performance of clip-level and frame-level prediction in Section 5.2. 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. These variants help us gain insights into the properties of the frame-level model.

5.1 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 show F1 score. In this paper, 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.

5.2 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. The thresholds are derived from MedleyDB. As Table 1a shows, using upscaling and using patching for up-sampling have little difference. As patching is computationally more demanding, we will report only the results using upscaling 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 [7], which achieving 0.898 average AUC. We also list the result using all 188 tags in Table 1b. The clip-level observations with 50 top tags also generally hold with 188 top tags. For space limit, we will report only the results with 50 top tags regarding clip-level hereafter.

5.3 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 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.

5.4 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. Most of the models with accumulation outperform the one 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 1a in some case.

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

To visually see the effect of accumulation, we show in Figure 3 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.

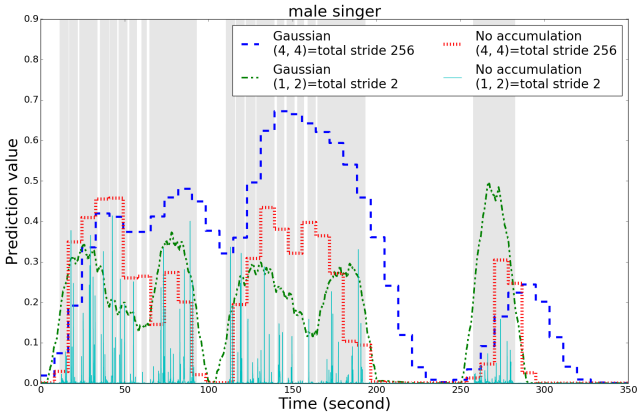


Figure 3: Smoothing effect of accumulation layers for a music piece from MedlyDB. We show the frame-level predictions of four models with final max-pooling. The micro F1 scores achieved by the four settings (Gaussian, stride 2), (Gaussian, stride 256), (No accumulation, stride 2) and (No accumulation, stride 256) are 0.658, 0.601, 0.158, and 0.601, respectively. This is the music piece entitled Alexander-Ross_GoodbyeBolero in the MedlyDB dataset.

Table 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		Rrecall		F1	
			Micro	Macro	Micro	Macro	Micro	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

Table 3: Accumulation layers. The models either use no accumulation, use accumulation with Gaussian of fixed σs, or use accumulation with Gaussian of adaptive σs. 50 classes for clip-level and 9 classes for frame-level

Final pooling	Early conv.	Accum. (Gaussian)	Frame level F1		Clip level Average AUC	
			Per-class	Per-clip	Micro	Macro
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 4: Effect of multi-scale inputs. Inputs of 1 scale, 3 scales, or 6 scales are experimented.

Early conv.	# of inputs	F1		Average AUC	
		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**

5.5 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 4 that the multi-scale inputs yield better results than input of a single scale. 6 scales yield the best results at either clip-level or frame-level, so we will use 6 scales in all the remaining experiments. We only show the results with adaptive Gaussian in Table 4 due to space limit, but the pattern also holds with fixed Gaussian.

5.6 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 [1] 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 [6]: "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."

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 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 get worse as the total stride increases. The pattern of the performance without Gaussian follows our observation in the Section 5.4 that frame-level predictions would be more difficult without accumulation.

5.7 Effect of Final Pooling Functions

We tested two pooling functions for the final layer, mean-pooling and max-pooling. From Table 5, there is no obvious advantage over the other using max and mean pooling. However, in principle, max pooling should be used if the training data have different durations to have comparable outputs.

5.8 Learned Parameters of Gaussian Filters

We show in Figure 4 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 5.2. 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 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 4 with green dashed line.

Even though they are linearly correlated to some degree, the number of training data is not the only factor affecting

Table 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.

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

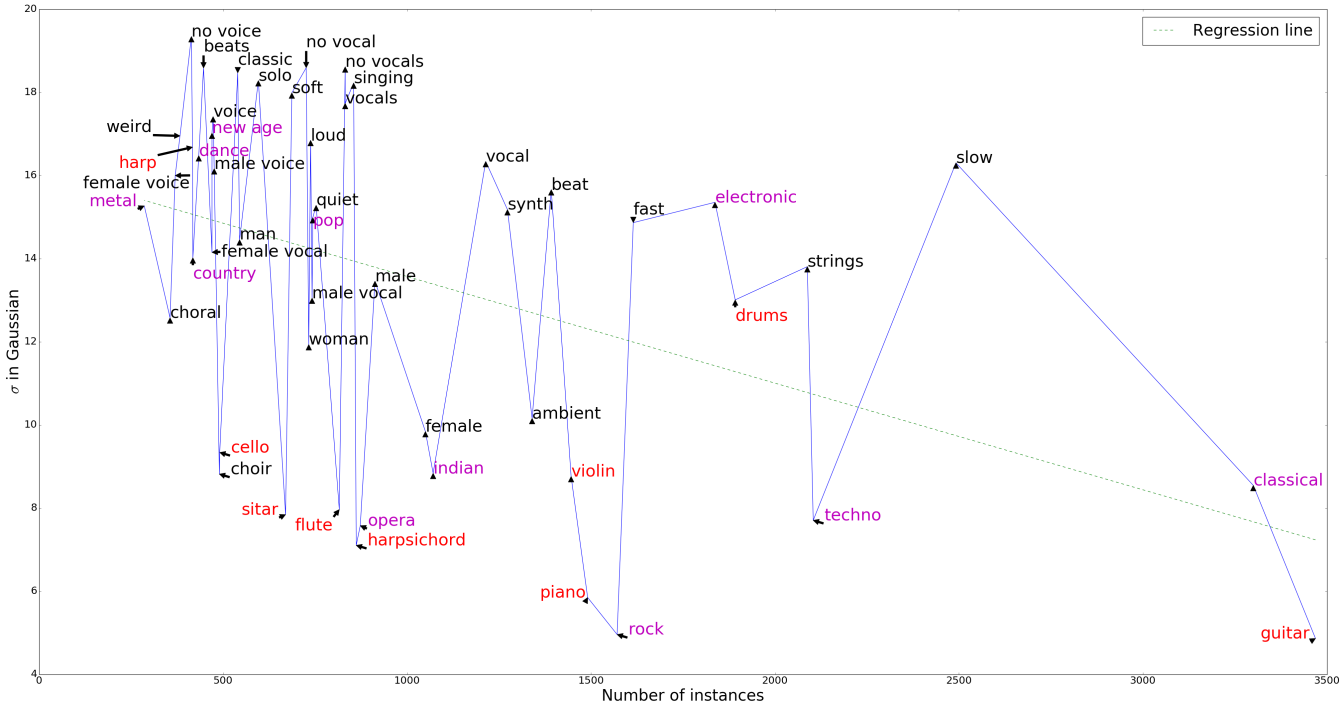


Figure 4: The standard deviation σ in Gaussian filter. x-axis is the number of instances in the training data, and y-axis is the quantity of σ of the classes after training. Instrument tags are in red color, and genre tags are in purple. The green dashed line is the linear regression line of the points on the image.

σs. The labels in red color are instruments, and the labels in purple color are genres. We can see that the σs of instruments are in general smaller than the σs of genres. This verifies our intuition that the instruments require smaller durations of time for recognition while genres are more complex so require larger durations for recognition. 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

Table 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

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 suggests that the patterns are not random.

5.9 Comparing with Frame-to-Frame training

We compare our model with a model trained directly with frame-level annotations from MedleyDB. We build neural networks (CNN with pooling_window=1) 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 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, we have divided the MedleyDB data in a more strict way such that the artists in the training set and the artists in the test set do not overlapping. 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 clip-to-frame training scheme we propose could be a more realistic solution to this problem.

6. VISUALIZATION OF THE FRAME-LEVEL PREDICTIONS

In this section, we visualize the frame-level predictions from the best frame-level model to subjectively evaluate the frame-level prediction. We show examples from the test data of the two datasets. The pre-accumulation layer in the figures refers to the layer right before the accumulation layer.

6.1 MedleyDB

Examples are shown in Figure 5. The examples in Figure 5a and 5b have good frame-level predictions and the threshold cuts at roughly the right place. The pre-accumulation output in Figure 5c looks reasonable, but the threshold cuts at a wrong place. The Figure 5d example 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 clean electric guitar. In MagnaTagATune, there is "electric guitar" tag but there is no distinction between distorted and clean electric guitar. 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 does not recognize the sound of clean electric guitar.

6.2 MagnaTagATune

We show one example from MagnaTagATune test data in Figure 6. Figure 6a is the predictions derived from the accumulation layer, and Figure 6b is the predictions from the pre-accumulation layer. By listening to the original clip, we can know that there is a man 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.

7. DISCUSSION AND CONCLUSION

In this paper, we have proposed a model 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 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 4, it is possible that the initial σs are too large. If the initial is too small or too large, it is difficult for classes with fewer data to learn good σs.

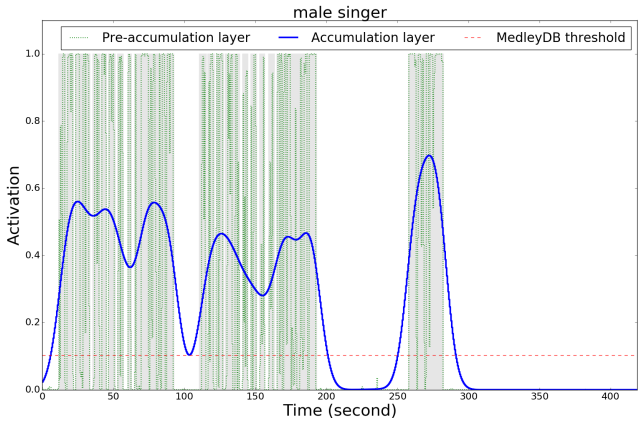
The number and diversity of the training dataset is an important issue. We have argued in Section 5.9 that the frame-to-frame training might not yield better result than our approach because the training data used by the frame-to-frame training is not diverse enough. In addition, we also observed in Section 6.1 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.

8. REFERENCES

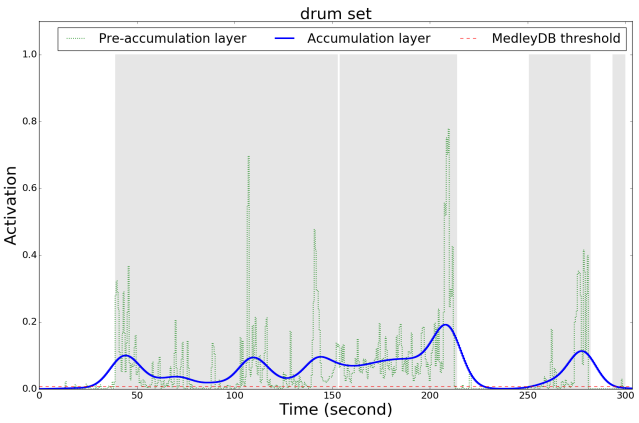
[1] O. Abdel-Hamid, L. Deng, and D. Yu. Exploring convolutional neural network structures and optimization techniques for speech recognition. In *Proc. INTERSPEECH*, pages 3366–3370, 2013.

[2] O. Abdel-Hamid, A. r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu. Convolutional neural networks for speech recognition. *IEEE/ACM TASLP*, 22(10):1533–1545, Oct 2014.

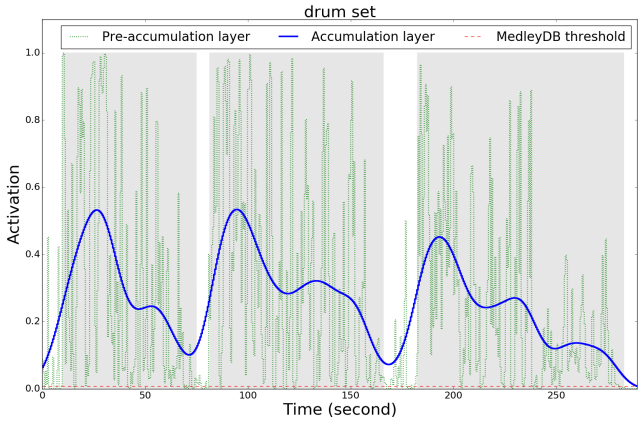
[3] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed



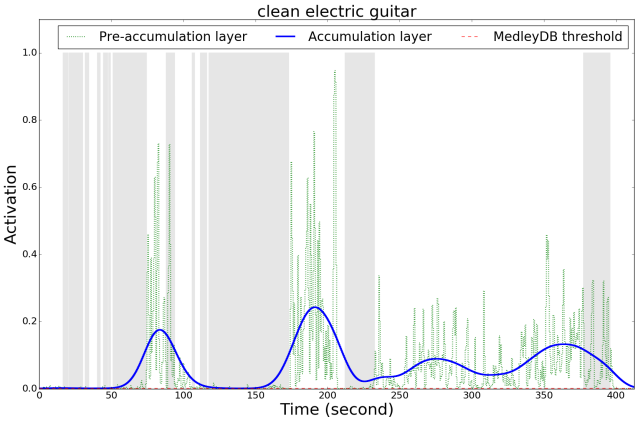
(a) Good. "male singer" from AlexanderRoss_GoodbyeBolero



(b) Good. "drum set" from TheScarletBrand_LesFleursDuMal



(c) Bad. "drum set" from StevenClark_Bounty



(d) Bad. "clean electric guitar" from Meaxic_YouListen

Figure 5: Visualization of frame-level predictions on MedleyDB test data. Two examples of good frame-level predictions and two of bad frame-level predictions are shown.

improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[4] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proc. the Python for Scientific Computing Conference (SciPy)*, June 2010. http://deeplearning.net/software/theano/.

[5] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello. Medleydb: A multitrack dataset for annotation-intensive mir research. In *Proc. ISMIR*, pages 155–160, 2014. http://medleydb.weebly.com.

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[7] S. Dieleman and B. Schrauwen. Multiscale approaches to music audio feature learning. In *Proc. ISMIR*, pages 116–121, 2013.

[8] S. Dieleman and B. Schrauwen. End-to-end learning for music audio. In *Proc. ICASSP*, pages 6964–6968, May 2014.

[9] D. Eck, P. Lamere, S. Green, and T. Bertin-Mahieux. Automatic generation of social tags for music recommendation. In *Proc. NIPS*, pages 1–8, 2007.

[10] S. Essid, G. Richard, and B. David. Instrument recognition in polyphonic music based on automatic taxonomies. *IEEE TASLP*, 14(1):68–80, jan 2006.

[11] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE TPAMI*, 32(9):1627–1645, 2010.

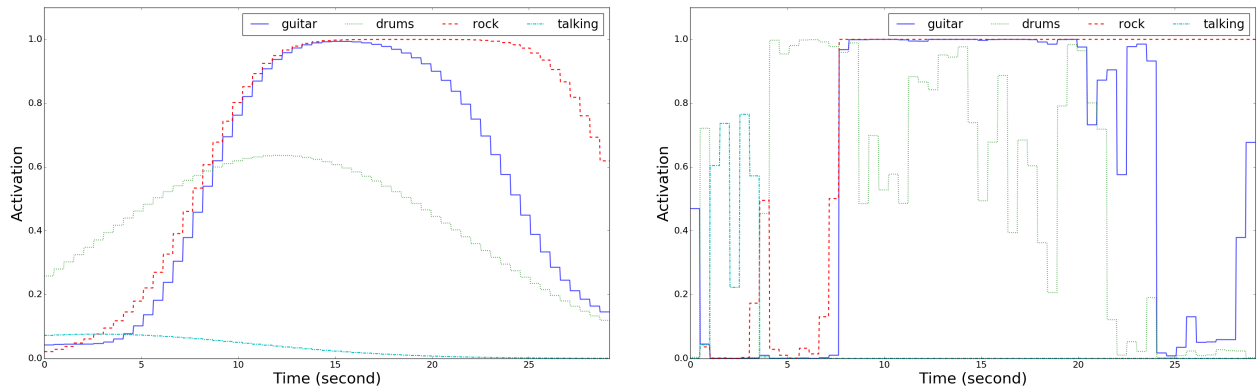
[12] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett. Darpa timit acoustic phonetic continuous speech corpus. 1993. *US Dept. of Commerce, NIST, Gaithersburg, USA*.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. CVPR*, pages 580–587, 2014.

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

[15] P. Hamel, S. Lemieux, Y. Bengio, and D. Eck. Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. In *Proc. ISMIR*, pages 729–734, 2011.

[16] M. Henaff, K. Jarrett, K. Kavukcuoglu, and



(a) Accumulation

(b) Pre-accumulation

Figure 6: 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.

Y. LeCun. Unsupervised learning of sparse features for scalable audio classification. In *Proc. ISMIR*, pages 681–686, 2011.

[17] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. 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.

[18] E. J. Humphrey, J. P. Bello, and Y. LeCun. Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In *Proc. ISMIR*, pages 403–408, 2012.

[19] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proc. CVPR*, 2014.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, pages 1097–1105, 2012.

[21] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie. Evaluation of algorithms using games: The case of music tagging. In *Proc. ISMIR*, 2009. http://mirg.city.ac.uk/codeapps/the-magnatagatune-dataset.

[22] M. Mandel, D. Eck, and Y. Bengio. Learning tags that vary within a song. *Proc. ISMIR*, pages 399–404, 2010.

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

[24] M. I. Mandel, R. Pascanu, D. Eck, Y. Bengio, L. M. Aiello, R. Schifanella, and F. Menczer. Contextual tag inference. *ACM TOMCCAP*, 7S(1):1–18, oct 2011.

[25] J. Masci, A. Giusti, D. Ciresan, G. Fricout, and J. Schmidhuber. A fast learning algorithm for image segmentation with max-pooling convolutional networks. In *Proc. ICIP*, pages 2713–2717, 2013.

[26] B. McFee, M. McVicar, C. Raffel, D. Liang, O. Nieto, E. Battenberg, J. Moore, D. Ellis, R. Yamamoto, R. Bittner, D. Repetto, P. Viktorin, J. F. Santos, and A. Holovaty. librosa: 0.4.1, Oct. 2015.

[27] J. Nam, J. Herrera, M. Slaney, and J. O. Smith. Learning sparse feature representations for music annotation and retrieval. In *Proc. ISMIR*, pages 565–570, 2012.

[28] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proc. CVPR*, pages 685–694, 2015.

[29] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proc. ICCV*, pages 1742–1750, 2015.

[30] G. Parascandolo, H. Huttunen, and T. Virtanen. Recurrent Neural Networks for Polyphonic Sound Event Detection In Real Life Recordings. In *Proc. ICASSP*, 2016.

[31] D. Pathak, E. Shelhamer, J. Long, and T. Darrell. Fully convolutional multi-class multiple instance learning. *arXiv preprint arXiv:1412.7144*, 2014.

[32] P. O. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In *Proc. ICML*, pages 82–90, 2014.

[33] P. O. Pinheiro and R. Collobert. From image-level to pixel-level labeling with convolutional networks. In *Proc. CVPR*, pages 1713–1721, 2015.

[34] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Proc. NIPS*, pages 568–576. 2014.

[35] D. Tingle, Y. E. Kim, and D. Turnbull. Exploring automatic music annotation with acoustically-objective tags. In *Proc. ICMR*, pages 55–61, 2010.

[36] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE TASLP*, 16(2):467–476, feb 2008.

[37] S.-Y. Wang, J.-C. W. Y.-H. Yang, and H.-M. Wang. Towards time-varying music auto-tagging based on cal500 expansion. In *Proc. ICME*, pages 1–6, 2014.

[38] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative cnn video representation for event detection. In *Proc. CVPR*, 2015.