High Precision Sound Event Detection based on Transfer Learning using Transposed Convolutions and Feature Pyramid Network

1st Shunyan Luo
Amazon
shunyl@amazon.com

2nd Yarong Feng
Amazon
yarongf@amazon.com

3st Zongyi(Joe) Liu
Amazon
joeliu@amazon.com

4nd Yuan Ling
Amazon
yualing@amazon.com

4nd Shujing Dong
Amazon
shujdong@amazon.com

5st Bruce Ferry
Amazon
bferry@amazon.com

***Abstract*—We introduce two models for high precision sound event detection leveraging transfer learning. The sound events we detect include "speech", "music", and "chime". Both models consist of a CNN backbone pre-trained using AudioSet for audio classification. To get high precision detection results, the first model employs transposed convolutional layers as the detection head, while the second model uses Feature Pyramid Network(FPN) as the detection head. Experimental results show 98.8% accuracy and 98.6% F1 score on a private test set, from the one using FPN. Both models outperform a two-stage model using LSTM, various model ensembles, and a pre-trained neural network model for audio classification.**

***Index Terms*—sound event detection, CNN, transposed convolution, feature pyramid network, audio signal processing**

I. Introduction

The problem we consider in this paper involves audio recordings of the dialogue between a voice assistant (such as Alexa in Echo Dot) and a "user". The audio recordings are collected in controlled lab experiments[1]. In an experiment, synthesized wake word(such as "Alexa") is first played through a speaker to the device(on which a voice assistant is installed) to wake it up, then a synthesized question(such as "play wheels on the bus") is played to the device. Afterwards, the device would respond by answering the question, performing the specific action, or playing the required content. Using a microphone placed next to both the device and the speaker, the first $20 \sim 60$ seconds of the dialogue is recorded(including wake word, question, and device response). Fig. 1 in [11] shows the lab hardware setup and Fig. 1 shows the waveforms of two audio recordings produced in these experiments.

In our case, such a dialogue usually follows this pattern: wake word(user) $\rightarrow$ question/request(user) $\rightarrow$ acknowledgement of receiving the question(voice assistant) $\rightarrow$ actual answer/content(voice assistant). For example, the first plot in Fig. 1 corresponds to the following dialogue:

- User: Alexa, add bananas to my shopping list.
- Alexa: You already have bananas on your shopping list.

and the second one corresponds to the following:

- User: Alexa, play wheels on the bus.
- Alexa: The wheels on the bus go round and round by the countdown Kids on Amazon music.
- Alexa: (music playing).

[1]The user's voice is synthesized. There is no recording of any actual conversation between customer and voice assistant involved.

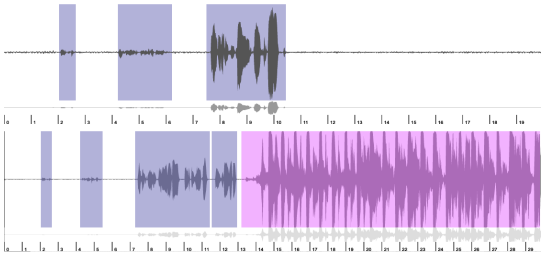

Fig. 1. Waveforms of two example dialogues. Blue blocks are speech components. Pink blocks are music components.

To better understand and analyze the audio recordings generated from these experiments, we would like to locate and classify each component of the dialogue. By locating a component, we mean providing the start and end times of the component, also known as onset and offset, such as $[2.30s, 7.65s]$. By classifying a component, we mean providing a label for it, with value in ["speech", "music", "chime"]. Silent pieces in the dialogue do not need to be identified. For the problem described in this paper, we do not differentiate between the voice of the user and that of the voice assistant, and consider both as "speech". The class "chime" refers to the specific sounds played by a voice assistant as acknowledgement for receiving a question or finishing a task.

As an example, the desired output for the bottom waveform in Fig. 1 would be:

Listing 1. example output of an audio recording

```
[{class: speech, start: 2.02, end: 2.59},
 {class: speech, start: 4.21, end: 5.50},
 {class: speech, start: 7.32, end: 11.34},
 {class: speech, start: 11.55, end: 13.01},
 {class: music, start: 13.20, end: 30.00}]
```

With components of a dialogue located and classified, we can evaluate a voice assistant's performance, or compare across different voice assistants/devices, over a wide range of metrics, such as response latency, media latency, etc. However, in order to draw sound conclusions regarding these latency metrics, measurement with high accuracy is a prerequisite. It is because: 1) many latency metrics tend to have small values, and 2) even for those metrics that are large in magnitude, when we make comparisons the difference is usually very small. As an example, Table I shows the response latency of a voice assistant from two devices. Roughly speaking, if a 10% margin of error is allowed, then to accurately evaluate the response latency of device A, we need to locate each component in the audio recording to be within 162ms from the ground truth. If we want to measure the difference in response latency between these devices, it requires each component to be located within 25ms$(1.87s - 1.62s)$ from the ground truth. Such high requirement on measurement accuracy calls for algorithms with high precision.

TABLE I
RESPONSE LATENCY COLLECTED FROM LAB EXPERIMENTS.

Metric	device A	device B
response latency P50 (second)	1.62	1.87

A. Related work

Voice activity detection(VAD) [1] is a common task in audio signal processing. Sound event detection(SED) [2] takes one step further, and tries to not only detect voice activities but also to locate and describe them. Depending on whether there are overlapping sound events, there are two types of SED tasks: monophonic where there is no overlapping sound events, and polyphonic where there are overlapping sound events. SED for real-world audio is usually polyphonic [3]. Another characteristic of SED for real-world audio is that, the audio data usually do not have high signal-to-noise ratio(SNR) [4]. While most SED algorithms deal with real-world audio, our use case differs from them in that the audio recordings are collected in well-controlled lab environment(high SNR), and there is no overlapping sound events(i.e., monophonic). However, as stated in Section I, our use case requires sound events be detected with high precision, making the problem challenging in a different dimension.

As SED result is usually more complex than that of audio tagging(especially for polyphonic SED), objective evaluation of SED performance is itself an ongoing research topic [5]. We follow the methodology in [5] to evaluate our models.

It's very challenging to acquire annotations needed for SED, because both the sound event labels and their onsets and offsets are needed. Recent work have been focusing on semi-supervised learning [6]. Thanks to the release of AudioSet [7], pre-trained deep learning models on large-scale data set [8] becomes available for audio tasks now.

The main contributions of our work are: 1) introduces CNN model architectures for high precision SED using transposed convolutions and FPN; 2) proves the efficacy of transfer learning from pre-trained CNN models for audio tagging to high precision SED.

II. DATA

We work with a dataset of roughly 700 raw audio recordings, collected in lab environment as described in Section I. Each audio recording has a duration of $20 \sim 60$ seconds, and sampling rate of 44.1kHz. For each audio, we manually labeled the ground truth, in the format as given in Listing 1. The dataset is randomly split into 80% training and 20% validation. Two separate test sets with ~ 150 audios are held out for final evaluation of model performance.

A. Data Preprocessing

The raw audios are first filtered by volume. Only audios for which the volume of all sound events fall between $66dB$ and $86dB$ are kept, where the thresholds are determined based on the volume distribution of the training data. The volume of a sound event is defined as $20 * \log_{10} A$, where A is the $90th$ percentile of the set of all local maxima in its amplitude.

Filtering by volume is necessary for our use case, as the raw audios come from different labs with (potentially) different hardware setup. In some labs, the volume could be so loud that it saturates the microphone, while in others so low that it's almost mixed up with background noise. Moreover, the volume in these labs could change frequently due to software update, hardware setup adjustment, etc. If we rely on a specific volume pattern in the training data, the model's performance will degrade significantly once that pattern changes, which is common as explained.

Each raw audio is then cut into several 10-second clips, which are partially overlapped with a hop size of 1 second. Doing so will not only speed up training by leveraging batch processing, but also works as a data augmentation technique.

B. Temporal Resolution in Ground Truth Representation

For audio tagging task, the ground truth is usually one or several labels for the entire audio, and is straightforward to generate. However, for SED tasks, the ground truth annotation usually exist in form similar to Listing 1, and it is not trivial to convert these free-form text labels to mathematical ones that are recognizable by models. A common way to do it is to first map the onset and offset timestamps of each sound event(for example $(2.02s, 2.59s)$), to the index of raw sampled data points in the audio vector(in this case $(89,082, 114,219)$ as we use sampling rate of 44.1kHz), which results in a vector of ground truth labels with the same length as the audio vector. The ground truth labels for raw data points are then aggregated

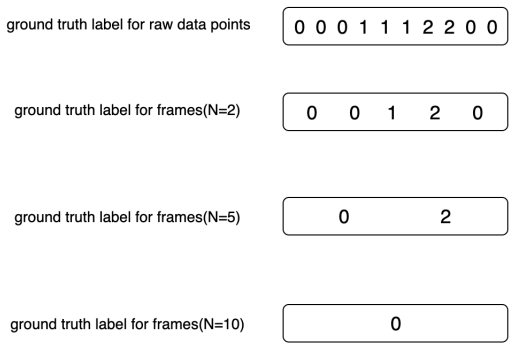

Fig. 2. Illustration of generating frame-wise ground truth labels. Ground truth as free-form text label: [{class: speech, start: 3, end: 5}, {class: music, start: 6, end: 7}]. Assuming sampling rate is 1Hz.

VGGish [1]	CNN6	CNN10	CNN14
Log-mel spectrogram 96 frames × 64 mel bins	Log-mel spectrogram 1000 frames × 64 mel bins		
3×3 @ 64 ReLU	5×5 @ 64 BN, ReLU	$\begin{pmatrix} 3 \times 3 \text{ @ } 64 \\ \text{BN, ReLU} \end{pmatrix} \times 2$	$\begin{pmatrix} 3 \times 3 \text{ @ } 64 \\ \text{BN, ReLU} \end{pmatrix} \times 2$
MP 2×2	Pooling 2×2		
3×3 @ 128 ReLU	5×5 @ 128 BN, ReLU	$\begin{pmatrix} 3 \times 3 \text{ @ } 128 \\ \text{BN, ReLU} \end{pmatrix} \times 2$	$\begin{pmatrix} 3 \times 3 \text{ @ } 128 \\ \text{BN, ReLU} \end{pmatrix} \times 2$
MP 2×2	Pooling 2×2		
$\begin{pmatrix} 3 \times 3 \text{ @ } 256 \\ \text{ReLU} \end{pmatrix} \times 2$	5×5 @ 256 BN, ReLU	$\begin{pmatrix} 3 \times 3 \text{ @ } 256 \\ \text{BN, ReLU} \end{pmatrix} \times 2$	$\begin{pmatrix} 3 \times 3 \text{ @ } 256 \\ \text{BN, ReLU} \end{pmatrix} \times 2$
MP 2×2	Pooling 2×2		
$\begin{pmatrix} 3 \times 3 \text{ @ } 512 \\ \text{ReLU} \end{pmatrix} \times 2$	5×5 @ 512 BN, ReLU	$\begin{pmatrix} 3 \times 3 \text{ @ } 512 \\ \text{BN, ReLU} \end{pmatrix} \times 2$	$\begin{pmatrix} 3 \times 3 \text{ @ } 512 \\ \text{BN, ReLU} \end{pmatrix} \times 2$
MP 2×2 Flatten	Global pooling		Pooling 2×2
FC 4096 ReLU $\times 2$	FC 512, ReLU		$\begin{pmatrix} 3 \times 3 \text{ @ } 1024 \\ \text{BN, ReLU} \end{pmatrix} \times 2$
FC 527, Sigmoid	FC 527, Sigmoid		Pooling 2×2
			$\begin{pmatrix} 3 \times 3 \text{ @ } 2048 \\ \text{BN, ReLU} \end{pmatrix} \times 2$
			Global pooling
			FC 2048, ReLU
			FC 527, Sigmoid

Fig. 3. Detailed CNN architecture from Table I in [8].

into frames using majority vote, where each frame represents N consecutive data points. See Fig. 2 for an illustration.

When aggregating raw data points into frames, the choice of N depends on the desired precision. For instance, with $N = 441$ each frame corresponds to 10ms in time, while with $N = 4410$ each frame corresponds to 100ms in time. In other words, the larger N is, the longer in time each frame covers, and the lower the temporal precision is for the frame-wise ground truth labels. In the example shown in Fig. 2, using $N = 2$ retains both the speech and music segments, using $N = 5$ keeps only the music segment, and using $N = 10$ results in both sound events being lost in the frame-wise labels.

We define the temporal resolution of a ground truth representation as the frames per second(FPS) it contains. As explained in Section I, our use case requires high precision and high temporal resolution. In particular, our goal is to build SED algorithms with at least 20FPS. Alternatively put, each frame in the output should cover at most 50ms in time.

III. Model Architecture

A. Backbone

We use the CNN14 model pre-trained on AudioSet in [8] up to the final fully connected layer as the backbone. Detailed architecture of it is shown in Fig. 3. The backbone has 6 convolutional blocks, each consists of 2 convolutional layers with a kernel size of 3 × 3. Batch normalization is applied between each convolutional layer, and the ReLU nonlinearity is used. Average pooling of size 2 × 2 is applied to each convolutional block(1 x 1 for the last one) for downsampling. Before convolutional blocks, the backbone converts the input from 1-d to 2-d by extracting the log mel spectrogram from the audio, which is typical in audio signal processing.

Note that in the original CNN14 model, a 10-second audio clip(with sampling rate $32k$Hz and hop size 320 data points) is first transformed into 1000 frames x 64 mel bins, then downsampled by half by each of the following convolutional block(except for the last one). After the first fully connected layer(which is the last layer in the backbone), input shape has been transformed to $(32, 2048)$, where 32 is the number of frames and corresponds to time , and 2048 is the number of channels and corresponds to the set of audio tags. Because the original CNN14 model is designed for audio tagging, downsampling is not only mathematically convenient, but also necessary in order to extract high level semantic information from the input. However, it does not meet the need of high precision SED, because the FPS is only $32/10 = 3.2$, far from the requirement stated in Section II-B.

A naïve solution is proposed in [8] to increase the temporal resolution of the model output by simple interpolation. For instance, if the model output has 5 FPS but the requirement is 20 FPS, then each frame in the model output is repeated 4 times to reach 20 FPS. Apparently this solution by simple interpolation only satisfies the temporal resolution requirement on the surface, but does not add any more information to the model output. We introduce two ways in the following sections that can increase the temporal resolution of the backbone output meaningfully using learnable parameters.

B. Transposed Convolution

Unlike regular convolutions, which usually map the input from a high-dimensional space to a low-dimensional space, transposed convolutions [9] are used when the opposite is needed: mapping the input from a low-dimensional space to a high-dimensional space. It also shares the benefits of a normal convolution: weight sharing, and learnable parameters. In our case, in order to increase the temporal resolution of the backbone output, we attach two one-dimensional transposed convolutional layers to it. Each layer has a stride of 4 and padding of 2, with kernel size 4 x 4. The full architecture is depicted in Fig. 4. The resulted temporal resolution is 668 frames for a 10-second clip, or 66.8 FPS.

C. Feature Pyramid Network

The drawback of using transposed convolution to increase temporal resolution is that it is purely based on the output of the last layer in the backbone, which usually contains only high-level features and little low-level features. When

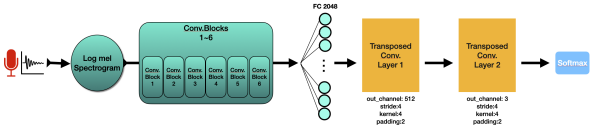

Fig. 4. Full architecture of CNN14 backbone + transposed convolutions.

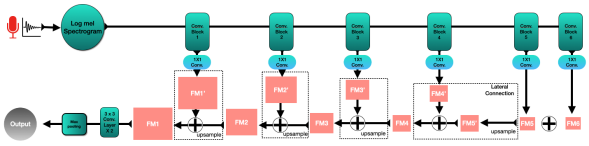

Fig. 5. Full architecture of CNN14 backbone + feature pyramid network. "FMn" stands for "feature map n".

the downsampling ratio is high(i.e., if use more convolutional blocks in the backbone), it is very challenging for the transposed convolutional layers to recover the low-level information lost in the convolutional process. When the downsampling ratio is low(i.e., if use fewer convolutional blocks in the backbone), on the other hand, input to the transposed convolutional layers may still be noisy and less efficient for detecting sound events globally. Feature Pyramid Network(FPN) [10] solves the problem by combing high-level features with low-level features, and leveraging both to produce the desired output.

The idea behind FPN is simple: given the outputs(feature maps) of two chained convolutional layers(e.g., conv1 $\rightarrow$ conv2), first upsample the feature map of conv2 so that its shape match that of the feature map of conv1, then combine them by element-wise addition, multiplication, concatenation, or other arithmetics. If using element-wise arithmetic, then each feature map should first pass through a 1 x 1 convolutional layer so as to equalize their number of channels. Upsampling can be done deterministically by simple interpolation, or be made part of the learning process with learnable parameters using ways such as transposed convolution. This process of upsampling and combination is called a lateral connection. Several lateral connections can be chained together in order to combine a series of features maps, at different levels, hence forming a feature pyramid.

The full architecture of our proposed CNN + FPN model is shown in Fig. 5. Output of each of the convolutional blocks first goes through a 1 x 1 convolutional layer to equalize number of channels, then passes through a series of lateral connections to form the feature pyramid. We use simple interpolation for upsampling and element-wise addition for feature map combination. Two additional 3 x 3 convolutional layers and a max pooling layer are appended to the feature pyramid, before the output layer.

FPN is originally designed for object detection at different scales, so several predictions are done in the model, one after each lateral connection. This is best depicted using Figure 1 in [10]. For our use case, we only make one prediction after the last lateral connection, because the goal is to detect sound events with high precision, or in other words, at the finest level of the feature pyramid. The resulted temporal resolution is 689 frames for a 10-second clip, or 68.9 FPS.

IV. Training

As explained in Section II, we cut the raw audio recordings as well as their annotations to 10-second clips, for model training. No other data augmentation technique is used. We use a sampling rate of 44.1kHz for the audio. To extract the spectrogram, we use a hop size of 320 samples, and Hanning window with length 1024 samples. To construct the log mel spectrogram, we apply 64 mel filter banks with lower and upper cut-off frequencies at 50Hz and 14kHz. We use an Adam optimizer with initial learning rate of 0.01, which is then multiplied by 0.1 for every 10 epochs. Both models are trained for 100 epochs with a batch size of 32, using cross entropy as the loss function. The model with the best validation accuracy among all epochs is taken to be the final trained model.

V. Inference

Both model architectures are designed in such a way that at inference time, they can take audio of arbitrary length as input, as long as the sampling rate is 44.1kHz, and output sound event detection results at the pre-defined temporal resolution. With the raw frame-wise predictions, an additional step needs to be taken to convert them to sound events as in Listing 1. A simple way to do that is to consider each continuous portion of frames with the same predicted label as one sound event. However, this method is very sensitive to false positives, i.e., silent frames predicted to be sound events. To further smooth out the predictions, we apply a simple post-processing logic by: 1) merging two adjacent sound events if the gap between them is $<$ 100 ms, 2) removing isolated sound events shorter than 300 ms, and 3) relabeling sound events shorter than 300 ms adjacent to longer ones.

Post processing is optional, but we find it necessary for our use case to get robust detection results. Note that the threshold values are chosen by experimenting on our test set, and should be re-evaluated (e.g., by cross validation) for other use cases.

VI. Evaluation

No overlapping sound events are allowed in our data generation process, so the SED task we handle is monophonic. Moreover, as we choose a discrete ground truth representation(although with high temporal resolution) using audio frames, it is natural to measure frame-level performance as for a multi-class and single-label classification problem. However, good frame-level performance is only a necessary but not sufficient condition for good event-level performance (i.e., estimating the start and end time of each sound event accurately). Prediction error on a key frame may only affect frame-level accuracy by a little, but could make the associated sound event location estimation be off by a lot. Therefore, it is critical to choose additional metrics that measure event-level performance properly as well.

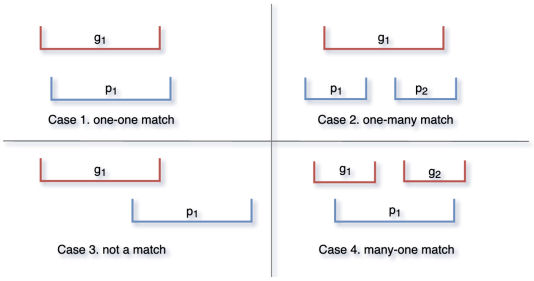

Fig. 6. Illustration of matching predicted events with ground truth.

In particular, we follow [5] and use its toolkit *sed_eval* to evaluate the frame-level performance(it's called segment-level in the toolkit and we shall follow this naming convention going forward). Although the toolkit is designed for polyphonic SED, it also works for monophonic results. For event-level performance, we use custom built metrics defined in [11].

A. Segment-level metrics

Segment-level metrics compare prediction with ground truth in short time segments. If the short time segments have the same duration as a frame, then these metrics reduce to frame-level metrics. We choose time segments of 20 ms and 1 s, respectively. False positives(FP), false negatives(FN), and true positives(TP) are counted, and accuracy, precision, recall, and F1 score are computed.

B. Event-level metrics

Event-level metrics compare predictions to ground truth event by event. However, we have noticed that the ground truth annotations in our dataset contain some level of subjectivity. For instance, in the ground truth annotation, a sound event sometimes can contain short gaps between words or sentences longer than the duration of a frame, which is at most 50 ms. At inference time, these gaps may be treated as background and not included in any detected sound event. As an example, the third and forth color blocks in the bottom plot of Fig. 1, may correspond to the same sound event in ground truth. Cases like this will distort the event-level metrics defined in [5], and may lead to problematic conclusions on the model performance. To properly measure event-level performance, we use the metrics from [11], which are based on a more robust matching mechanism between the predicted events and ground truth events. As depicted in case 2 and case 4 of Fig. 6, multiple predicted events could be matched with the same ground truth event, and vice versa. The event-level metrics are computed for each matched group. This way, we minimize the impact on the computed metrics from the subjectivity embedded in ground truth annotation regarding short gaps. Specifically, we use mean IoU, mean front miss, FN, and FP.

VII. Results

There is scarcity of both algorithm and dataset for high-precision SED. To the best of the authors' knowledge, there is no off-the-shelf algorithm for high-precision SED that suits our need. Moreover, almost all benchmark datasets contain audios in the real-world setting or synthetic audios, both differ greatly from audios collected in a lab environment. As a result, we choose to evaluate the models using two private datasets: test set I and II. Test set I contains 128 lab-collected raw audio recordings that are similar to those used in training. Test set II has 25 raw audio recordings, which we know the current production model has made mistakes on. In other words, metrics on test set I can show the overall performance of a model, while metrics on test set II can reveal whether a model can beat the status quo. Neither test set I nor test set II audios have been filtered by volume.

We compare our models' performance against three alternatives: 1) the current model in production(abbreviated to PROD), which is two-stage and based on LSTM(see details in [11]); 2) the current model in production, but with additional sound event label verification using the CNN14 model in [8](abbreviated to PROD+CNN14); 3) the CNN14 model in [8] only, used for SED with simple interpolation to increase temporal resolution(abbreviated to CNN14+INTERPOLATE). 2) is chosen as an alternative based on the observation that the current production model tends to confuse music sound events with speech, but the onset and offset of each event are pretty accurate. We hope that the CNN14 model is able to fix the misclassification issue in this additional verification step. The two models we propose in this paper are abbreviated to CNN14+TRANS CONV, and CNN14+FPN, respectively.

Segment-level metrics are shown in Table II, Table III, Table IV using micro averages. Event-level metrics are reported by class in Table V and Table VI respectively.

On test set I, in terms of segment performance, CNN14+FPN and CNN14+TRANS CONV beats all other alternatives on all metrics, regardless of segment duration. Between the two, CNN14+FPN is slightly better for all metrics, except for precision. CNN14+INTERPOLATE has the worst performance. This is not surprising because the model is not fine-tuned at all. PROD+CNN14 shows big improvement over PROD, but still does not beat the proposed methods.

On test set II, the proposed methods are still the winner. Note the poor performance of PROD here, as this data set is chosen on purpose to consist only of PROD failures. Still, PROD+CNN14 is able to boost the performance by $\sim 30\%$, but still cannot catch up with the proposed methods. Even for the proposed methods, performance drops on test set II, reflecting the fact that these cases are challenging.

In terms of event-level metrics, the proposed methods have the best mean IoU, mean front miss, and FN, but tend to produce more false positives than other methods. This is consistent across sound event classes.

To shed more light on the specific errors each model tends to make, we compare the raw predictions from PROD and the two proposed models on the same audio input, in Fig. 7. In ground truth, the first three events are "speech" and the last one is "music". Consistent with the results in Table VI, PROD fails to classify the start(and the end) of the music segment correctly,

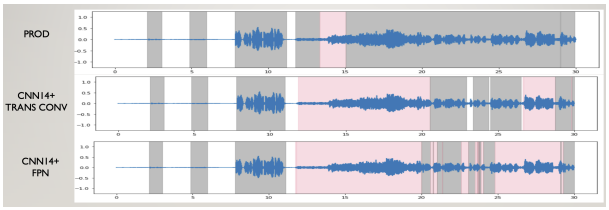

Fig. 7. Raw predictions from three methods on the same audio input.

resulting in high front miss, but it generates very few FP because the predicted events are relatively long. On the other hand, both proposed models get the music start correctly and accurately, corresponding to much lower front miss. However, both produce more false speech predictions in the middle of the music segment, and because these predicted events are shorter and disconnected, the number of false positives is much higher. This issue is more serious for CNN14+FPN, probably due to the fact that it incorporates more low-level features than transposed convolution. Note that the predictions in Fig. 7 are raw without post processing. We believe the post processing logic described in Section V can address most the issues present in this figure.

TABLE II
SEGMENT-LEVEL PERFORMANCE ON TEST SET I USING 1-SECOND SEGMENTS.

Method	**Test Set I**			
(1s segment)	accuracy	F1 score	precision	recall
PROD	95.8%	95%	94.9%	95.1%
PROD+CNN14	98%	97.7%	97.5%	97.8%
CNN14+INTERPOLATE	91.9%	90.4%	90%	90.9%
CNN14+TRANS CONV	98.6%	98.3%	98.2%	98.5%
CNN14+FPN	98.8%	98.6%	98.1%	99.2%

TABLE III
SEGMENT-LEVEL PERFORMANCE ON TEST SET I USING 20MS SEGMENTS.

Method	**Test Set I**			
(20ms segment)	accuracy	F1 score	precision	recall
PROD	95.8%	94.2%	94.1%	94.2%
PROD+CNN14	98%	97.2%	97.1%	97.2%
CNN14+INTERPOLATE	90.7%	90.4%	90%	82.2%
CNN14+TRANS CONV	98.3%	97.6%	97.8%	97.5%
CNN14+FPN	98.8%	98.3%	98.2%	98.3%

TABLE IV
SEGMENT-LEVEL PERFORMANCE ON TEST SET II USING 1-SECOND SEGMENTS.

Method	**Test Set II**			
(1s segment)	accuracy	F1 score	precision	recall
PROD	62.4%	55.2%	54.5%	55.9%
PROD+CNN14	93.7%	92.4%	91.6%	93.2%
CNN14+INTERPOLATE	88.1%	85.3%	87.7%	82.9%
CNN14+TRANS CONV	96.8%	96.2%	95.7%	96.6%
CNN14+FPN	97.3%	96.8%	95.3%	98.4%

TABLE V
EVENT-LEVEL PERFORMANCE ON TEST SET I FOR CLASS "SPEECH".

Method	**Speech**			
	IoU	front miss	FP	FN
PROD	0.94	0.04	24	2
CNN14+INTERPOLATE	0.72	0.32	22	0
CNN14+TRANS CONV	0.92	0.04	12	0
CNN14+FPN	0.95	0.03	27	0

TABLE VI
EVENT-LEVEL PERFORMANCE ON TEST SET I FOR CLASS "MUSIC".

Method	**Music**			
	IoU	front miss	FP	FN
PROD	0.91	0.28	1	4
CNN14+INTERPOLATE	0.76	1.22	6	3
CNN14+TRANS CONV	0.95	0.14	8	0
CNN14+FPN	0.97	0.11	14	0

VIII. CONCLUSION

We propose two CNN model architectures for high-precision sound event detection based on transfer learning. Both outperform the existing method and other alternatives on the test set. Although accurate, these high-precision SED algorithms tend to produce more false positives. The authors would like to explore other model architectures such as audio transformer to overcome this issue.

REFERENCES

[1] J. Ramirez, J. M. Górriz, and J. C. Segura, "Voice activity detection. fundamentals and speech recognition system robustness," *Robust speech recognition and understanding*, vol. 6, no. 9, pp. 1–22, 2007.

[2] S. Adavanne, A. Politis, J. Nikunen, and T. Virtanen, "Sound event localization and detection of overlapping sources using convolutional recurrent neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 1, pp. 34–48, 2018.

[3] A. Dang, T. H. Vu, and J.-C. Wang, "A survey of deep learning for polyphonic sound event detection," *2017 International Conference on Orange Technologies (ICOT)*, pp. 75–78, 2017.

[4] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," *2010 18th European Signal Processing Conference*, pp. 1267–1271, 2010.

[5] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, 2016.

[6] D. De Benito-Gorrón, D. Ramos, and D. T. Toledano, "A multi-resolution crnn-based approach for semi-supervised sound event detection in dcase 2020 challenge," *IEEE Access*, vol. 9, pp. 89 029–89 042, 2021.

[7] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[8] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, 2020.

[9] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," arXiv, 2018.

[10] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," arXiv, 2017.

[11] F. Yarong, L. Zongyi, L. Yuan, and F. Bruce, "A two-stage lstm based approach for voice activity detection with sound event classification," *International Conference on Consumer Electronics*, 1 2022.