




Article

You Only Hear Once: A YOLO-like Algorithm for Audio Segmentation and Sound Event Detection

Satvik Venkatesh ^{1,*} , David Moffat ^{1,2}  and Eduardo Reck Miranda ¹ 

¹ Interdisciplinary Centre for Computer Music Research, University of Plymouth, Plymouth PL4 8AA, UK; dmof@pml.ac.uk (D.M.); eduardo.miranda@plymouth.ac.uk (E.R.M.)

² Plymouth Marine Laboratory, Plymouth PL1 3DH, UK

* Correspondence: satvik.venkatesh@plymouth.ac.uk

Abstract: Audio segmentation and sound event detection are crucial topics in machine listening that aim to detect acoustic classes and their respective boundaries. It is useful for audio-content analysis, speech recognition, audio-indexing, and music information retrieval. In recent years, most research articles adopt segmentation-by-classification. This technique divides audio into small frames and individually performs classification on these frames. In this paper, we present a novel approach called You Only Hear Once (YOHO), which is inspired by the YOLO algorithm popularly adopted in Computer Vision. We convert the detection of acoustic boundaries into a regression problem instead of frame-based classification. This is done by having separate output neurons to detect the presence of an audio class and predict its start and end points. The relative improvement for F-measure of YOHO, compared to the state-of-the-art Convolutional Recurrent Neural Network, ranged from 1% to 6% across multiple datasets for audio segmentation and sound event detection. As the output of YOHO is more end-to-end and has fewer neurons to predict, the speed of inference is at least 6 times faster than segmentation-by-classification. In addition, as this approach predicts acoustic boundaries directly, the post-processing and smoothing is about 7 times faster.

Keywords: audio segmentation; sound event detection; you only look once; deep learning; regression; convolutional neural network; music-speech detection; convolutional recurrent neural network; radio

1. Introduction

Audio segmentation and sound event detection have similar goals—to detect acoustic classes and their respective boundaries within an audio stream. They provide information regarding the content of audio and the temporal occurrences of audio events. It is helpful for indexing audio archives, target-based distribution of media, and as a pre-processing step for speech recognition [1]. In addition, detecting audio events in real-time is beneficial for self-driving automobiles [2], surveillance [3], bioacoustic monitoring [4], and intelligent remixing [5].

The literature has commonly adopted two approaches to audio segmentation—(1) distance-based segmentation and (2) segmentation-by-classification [6]. The first approach directly finds regions of high acoustic change through Euclidean distance or Bayesian information criterion [7]. The method divides audio into segments based on the peaks of acoustic change. Subsequently, the audio classes within each of these segments are detected. However, recent research has generally adopted the second approach, which is segmentation-by-classification. It presents sound event detection as a supervised learning task. This approach divides an audio file into frames, typically in the range of 10–25 ms, and classifies each frame individually. Effectively, we detect the onset and offset of each audio event by classification.



Citation: Venkatesh, S.; Moffat, D.; Miranda, E.R. You Only Hear Once: A YOLO-like Algorithm for Audio Segmentation and Sound Event Detection. *Appl. Sci.* **2022**, *12*, 3293. <https://doi.org/10.3390/app12073293>

Academic Editor: Sławomir K. Zieliński

Received: 8 March 2022

Accepted: 22 March 2022

Published: 24 March 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Data to train a machine learning model for event detection require precise labels that mention the acoustic boundaries and classes. Annotating such datasets is a time-consuming and expensive process. Therefore, researchers have explored data-centric approaches such as artificial data synthesis to generate large-scale training data [8,9]. Furthermore, researchers have explored weak label and semi-supervised learning to tackle the scarcity of labelled data [10,11]. Datasets such as AudioSet [12] are annotated with weak labels, which indicates that a sound event is present in the audio clip, but does not specify the timing within the audio. Hershey et al. [13] emphasised the benefit of temporally strong labels to improve the performance of audio classifiers.

The architectures for audio segmentation have evolved from traditional machine learning models such as the Gaussian mixture model to deep neural networks. Bidirectional Long short-term memory (B-LSTM) networks have been effective in segmenting temporal data [14]. Lemaire et al. [15] showed that the non-causal temporal convolutional neural network was more effective than the B-LSTM. However, the Convolutional Recurrent Neural Network (CRNN) obtains state-of-the-art performance on many sound event detection datasets because it combines the advantage of 2D convolutions and recurrent layers [16,17].

There has been a growing interest in the community to adopt end-to-end deep learning for information retrieval from audio. Raw audio waveforms have been explored instead of features such as mel spectrograms for the input [18,19]. However, with regards to an end-to-end setup, there has been less attention given to the output of such networks. Traditionally, in segmentation-by-classification, the neural network classifies each audio frame. Subsequently, a post-processing step converts the neural network's output into human-readable labels. The disadvantage is that this post-processing is slow because each audio frame has to be serially processed. Therefore, in an ideal end-to-end setup, the neural network would output human-readable labels by directly predicting the boundaries of acoustic classes.

In order to output human-readable labels directly, sound event detection must be transformed from a classification problem to a regression problem. Phan et al. [20] proposed random regression forests for sound event detection and classification. Xu et al. [21] adopted a regression approach for speech enhancement. However, most studies in the literature adopt frame-based classification, where the neural network classifies each frame separately. In this study, we present a novel neural network architecture inspired by the You Only Look Once (YOLO) algorithm [22]. YOLO gained attention in the Computer Vision community for object detection. It transformed bounding box prediction from a classification problem to a regression one. Using this approach, it obtained speedups of around $3\times$ without compromising accuracy. We present a system called You Only Hear Once (YOHO) that predicts the boundaries of acoustic classes through regression.

YOLO has been adopted in the audio domain by visualising spectrograms as images. Zsebők et al. [23] adopted YOLO for automatic bird song and syllable segmentation. Segal et al. [24] presented a system called SpeechYOLO which treated audio fragments as objects. They adopted YOLO for keyword spotting tasks. Algabri et al. [25] investigated object detection techniques such as YOLO and CenterNet [26] for phoneme recognition. However, the novelty of the YOHO paradigm is that it converts frame-based classification into a regression problem by gradually reducing the temporal dimension through many convolutional layers. This makes the output of the network closer to human-readable labels, therefore reducing the need for post-processing. Separate neurons were used to detect the onset and offset of audio classes. We apply our system to audio segmentation and sound event detection tasks, where the literature has predominantly used frame-based classification. Furthermore, we present a multi-output system, which detects acoustic classes that can overlap with each other.

We evaluate the YOHO algorithm for multiple audio event detection tasks. First, we explore music-speech detection in broadcast signals. We also compare our results with state-of-

the-art algorithms on the Music Information Retrieval Evaluation eXchange (MIREX) competition dataset 2018 [27]. Second, we test our model on the TUT sound event detection dataset, which represents common sounds related to human presence and traffic. It was the dataset used in the Detection and Classification of Acoustic Scenes and Events (DCASE) competition 2017 [28]. Third, we evaluate our model on the Urban-SED dataset [9], which is a synthetic dataset for environmental audio. In all three cases, the YOHO algorithm performed better and faster than the CRNN. All the code associated with this project is available in this GitHub repository (<https://github.com/satvik-venkatesh/you-only-hear-once>, accessed on 2 March 2022).

2. You Only Hear Once (YOHO)

2.1. Motivation

In this paper, we intend to make the neural network output labels that are closer to human-readable labels. This way we make the pipeline more end-to-end. Figure 1 illustrates a comparison between segmentation-by-classification and the YOHO paradigm. For both paradigms, a mel spectrogram of shape 801×64 is fed as input. In segmentation-by-classification, each time step is classified as music, speech, both, or none. Subsequently, these classifications are converted to human readable labels. However, in YOHO, each block of 0.307 s is processed through regression. One neuron detects the presence of an acoustic class. If the class is present, one neuron predicts the start point of the class and one neuron detects the end point of the class. Subsequently, during post-processing, these blocks of 0.307 s are merged to form a final prediction. Using this technique, the number of time steps is reduced from 801 to 26, which makes the network significantly faster, generalise better, and more end-to-end. More details on the implementation are given in the below subsections.

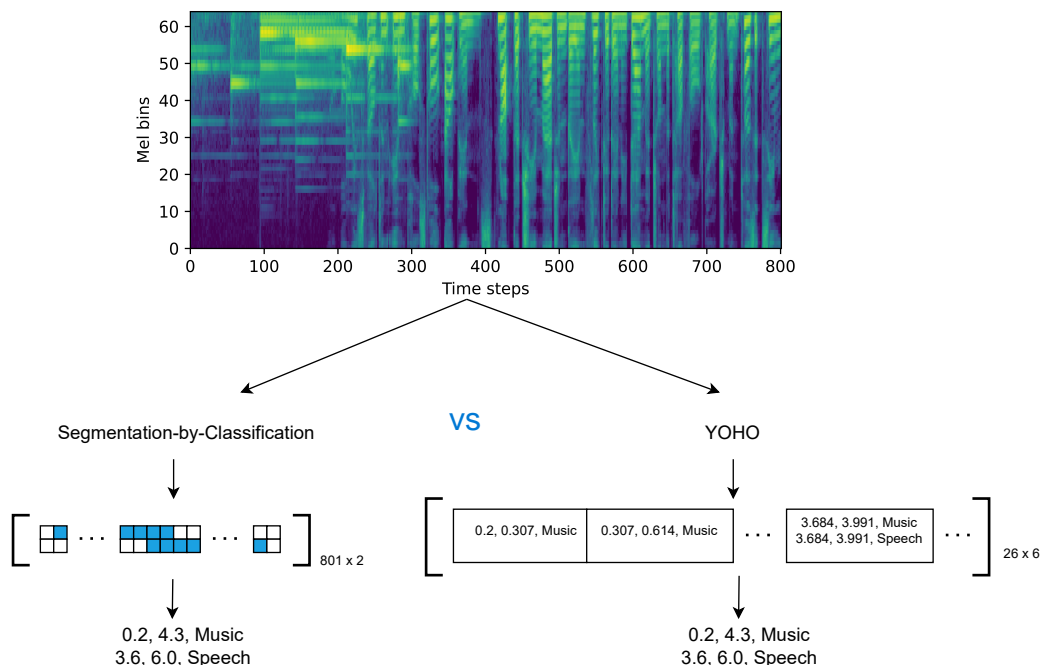


Figure 1. A comparison of segmentation-by-classification and YOHO.

2.2. Network Architecture

The network architectures used by different versions of YOLO [22,29] were large and not suitable for our smaller training datasets. Therefore, we adapted the MobileNet architecture

[30] for our task. We modified the final layers of MobileNet to realize the YOHO algorithm. MobileNet has also been employed for audio classification by YamNet [31], which only detects audio classes, but not their segmentation boundaries.

As shown in Table 1, YOHO is purely a convolutional neural network (CNN). We divide the table into two parts—the upper half comprising the original layers of the MobileNet architecture and the bottom half containing the layers that we have added. We use log-mel spectrograms as input features. The input dimension depends on the duration of the audio example and specifications of the mel spectrogram. Here, we explain the network for music-speech detection, whose input contains 801 time steps and 64 frequency bins. After reshaping the mel spectrogram to $801 \times 64 \times 1$, we perform a 2D convolution with a stride of 2. Hence, the time dimension and frequency dimension are reduced by half. The MobileNet architecture uses many depthwise-separable convolutions [32] with 3×3 filters followed by pointwise convolutions with 1×1 filters. All convolutions except the final layer were fitted with ReLU activations and batch normalization [33]. Each time we adopt a stride of 2, there is a reduction in the time and frequency dimensions. As shown in the lower half of Table 1, we gradually reduce the number of filters from 1024 to 256.

Table 1. The neural network architecture for YOHO. The upper half of the table comprises the original layers of MobileNet. The bottom half contains the layers that we have added. Conv2D and Conv1D stand for 2D and 1D convolutions, respectively. The convolutions use a stride of 1 unless mentioned otherwise and ‘dw’ stands for depthwise convolution.

	Layer Type	Filters	Shape/Stride	Output Shape
	Reshape	-	-	$801 \times 64 \times 1$
	Conv2D	32	$3 \times 3/2$	$401 \times 32 \times 32$
	Conv2D-dw	-	3×3	$401 \times 32 \times 32$
	Conv2D	64	1×1	$401 \times 32 \times 64$
	Conv2D-dw	-	$3 \times 3/2$	$201 \times 16 \times 64$
	Conv2D	128	1×1	$201 \times 16 \times 128$
	Conv2D-dw	-	3×3	$201 \times 16 \times 128$
	Conv2D	128	1×1	$201 \times 16 \times 128$
	Conv2D-dw	-	$3 \times 3/2$	$101 \times 8 \times 128$
	Conv2D	256	1×1	$101 \times 8 \times 256$
	Conv2D-dw	-	3×3	$101 \times 8 \times 256$
	Conv2D	256	1×1	$101 \times 8 \times 256$
	Conv2D-dw	-	$3 \times 3/2$	$51 \times 4 \times 256$
	Conv2D	512	1×1	$51 \times 4 \times 256$
5×	Conv2D-dw	-	3×3	$51 \times 4 \times 256$
	Conv2D	512	1×1	$51 \times 4 \times 256$
	Conv2D-dw	-	$3 \times 3/2$	$26 \times 2 \times 512$
	Conv2D	1024	1×1	$26 \times 2 \times 1024$
	Conv2D-dw	-	3×3	$26 \times 2 \times 1024$
	Conv2D	1024	1×1	$26 \times 2 \times 1024$
	Conv2D-dw	-	3×3	$26 \times 2 \times 1024$
	Conv2D	512	1×1	$26 \times 2 \times 512$
	Conv2D-dw	-	3×3	$26 \times 2 \times 512$
	Conv2D	256	1×1	$26 \times 2 \times 256$
	Conv2D-dw	-	3×3	$26 \times 2 \times 256$
	Conv2D	128	1×1	$26 \times 2 \times 128$
	Reshape	-	-	26×256
	Conv1D	6	1	26×6

Subsequently, we flatten the last two dimensions. The final layer is a 1D convolution with six filters. The output shape is 26×6 , where 26 stands for the number of time steps. This layer is similar to a convolutional implementation of sliding windows [34] along the time dimension. At each time step, the first neuron performs a binary classification that detects the presence of

an acoustic class. The second and third neurons perform regression for the start and endpoints for the respective acoustic class. Figure 2 illustrates the output layer of the YOHO algorithm.

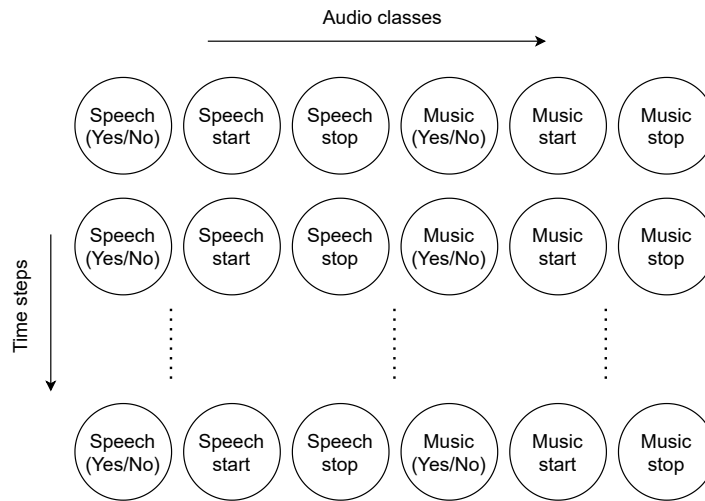


Figure 2. An illustration of the output layer of the YOHO algorithm. This network is for music-speech detection. To increase the number of audio classes, we add neurons along the horizontal axis.

In this context, we are dealing with two acoustic classes—music and speech. Therefore, the output has six neurons at each time step. For example, if the length of an audio example is 8 s, each time step in the output corresponds to 0.307 s because there are 26 divisions. We applied sigmoid activations for all neurons in the output layer. Hence, we normalized the regression outputs between 0 and 1. Moreover, even if the input shape of the neural network is different, for example 257×40 , the neural network and the parameters of convolutional layers still remain exactly the same. The only difference would be the output shape of the neural network, which depends on the number of time steps in the input and the number of unique audio classes in the output.

2.3. Loss Function

Generally, neural networks such as the CRNN that use segmentation-by-classification adopt binary cross-entropy as the loss function. As we modeled the problem as a regression one, we used the sum squared error. Equation (1) shows the loss function for each acoustic class c .

$$\mathcal{L}_c(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + (\hat{y}_3 - y_3)^2, & \text{if } y_1 = 1 \\ (\hat{y}_1 - y_1)^2, & \text{if } y_1 = 0 \end{cases} \quad (1)$$

where y and \hat{y} are the ground-truth and predictions respectively. $y_1 = 1$ if the acoustic class is present and $y_1 = 0$ if the class is absent. y_2 and y_3 , which are the start and endpoints for each acoustic class are considered only if $y_1 = 1$. In other words, $(\hat{y}_1 - y_1)^2$ corresponds to the classification loss and $(\hat{y}_2 - y_2)^2 + (\hat{y}_3 - y_3)^2$ corresponds to the regression loss. The total loss \mathcal{L} is summed across all acoustic classes.

2.4. Example of Labels

Table 2 shows an example of the output for the YOHO algorithm. The total length of the audio is 8 s. Within the example, Music occurs from 0.2 to 4.3 s and Speech occurs from 3.6 to 6.0 s. Note that each row in Table 2 corresponds to one time step, which is equal to 0.307 s. In

addition, the regression values are normalized from 0 to 1. For example, if music starts at 0.2 s, the value is divided by 0.307 to get 0.65 as shown in the first row of Table 2.

Table 2. An example of labels for the YOHO algorithm. Music occurs from 0.2 to 4.3 s and Speech occurs from 3.6 to 6.0 s. Note that start and stop values are considered only when the respective audio class is present. The dimensions of the output are 26×6 . Note that each time step/row in the table corresponds to 0.307 s. The start and stop values are normalised on the range of 0 to 1. For instance, in the first time step, music’s start point would be rescaled from 0.2 to 0.65.

Speech (Yes/No)	Speech Start	Speech Stop	Music (Yes/No)	Music Start	Music Stop
0	-	-	1	0.65	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
0	-	-	1	0.0	1.0
1	0.7	1.0	1	0.0	1.0
1	0.0	1.0	1	0.0	1.0
1	0.0	1.0	1	0.0	0.975
1	0.0	1.0	0	-	-
1	0.0	1.0	0	-	-
1	0.0	1.0	0	-	-
1	0.0	1.0	0	-	-
1	0.0	1.0	0	-	-
1	0.0	0.5	0	-	-
0	-	-	0	-	-
0	-	-	0	-	-
0	-	-	0	-	-
0	-	-	0	-	-
0	-	-	0	-	-
0	-	-	0	-	-

2.5. Other Details

We trained the network with the Adam optimizer, a learning rate of 0.001, a batch size of 32, and early stopping [35]. In some cases, we used L2 normalization, spatial dropout, and SpecAugment [36]. We used log-mel spectrograms as features for the neural network. The parameters of spectrograms were unique for each dataset. Section 3 contains the details for each case.

To evaluate the systems, we adopted the sed_eval toolbox [37], which is common in the literature for audio segmentation and sound event detection. The python toolbox is openly available (https://tut-arg.github.io/sed_eval/, accessed on 17 March 2022) and presents a convenient interface to calculate metrics such as overall F-measure, error rate, class-based F-measures, and so on. The specifications of segment-based metrics for experiments in this paper are mentioned along with the relevant results in Section 4.

2.6. Post-Processing

For music-speech detection, the output of the CRNN would be 801×2 , corresponding to 801 time steps and two acoustic classes. On the other hand, the output for the YOHO network is 26×6 . A post-processing step parses the output of the neural network to create human-readable labels. Subsequently, smoothing is performed over the output to eliminate the occurrence of spurious audio events. Two smoothing approaches are common in the literature—median filtering [14] and threshold-dependent smoothing [15]. We adopted the latter approach. In this technique, if the duration of the audio event is too short or if the silence between consecutive events of the same acoustic class is too short, we remove the occurrence.

For music-speech detection, the minimum silence between consecutive music events or consecutive speech events was set to 0.8 s. The minimum duration for a music event was set to 3.4 s and for a speech event was 0.8 s. For environmental sound event detection, if the silence between consecutive audio events of the same acoustic class was less than 1.0 s, it was smoothed. We did not set any threshold for the minimum duration of an audio event for this task.

2.7. Models for Comparison

In this sub-section, we present two additional models, which are slight deviations from the YOHO architecture—CNN and CRNN. The motivation behind these models is to investigate which aspects of YOHO are actually advantageous. The feature-extraction for all these models is the same, which make them directly comparable. The CNN model aims to create a segmentation-by-classification version of the YOHO architecture. As you can see in Table 1, some Conv2D-dw layers adopt a stride of [2, 2]. These strides were set to [1, 1] instead of [2, 2] and max-pooling was adopted to reduce the frequency dimension by half. This way, the time resolution of the network does not reduce through its depth. Note that using a stride of [1, 2] would have produced a similar effect of maintaining the time resolution and reducing the frequency resolution. However, TensorFlow currently does not support rectangular strides for depthwise convolutions and hence, we adopted max-pooling. The number of parameters in the CNN was 3.9 million, which is the same as the network for YOHO.

In the CRNN model, the first 13 layers were identical to the YOHO network. We skipped the convolutional layers where the number of filters became larger than 256 because the network became too large to fit into the RAM. Following the convolutional layers, we had two B-GRU layers with 80 units each. The number of parameters for the CRNN was 1.3 million, which is less than the YOHO network. Increasing the number of convolutional layers only worsened the performance of the CRNN. Therefore, it was optimal to have a CRNN with fewer parameters.

The output shape for the CNN and CRNN was 801×2 , performing binary classification for music and speech at each time step. We compared the performance of YOHO with these two additional models on the in-house test set for music-speech detection. We also compared the inference times of these models. A summary of the architectures for comparison can be found in Table 3.

Table 3. Models for comparison on the in-house test set for music-speech detection.

Model	Remarks
YOHO	The architecture is explained in Section 2.2.
CNN	[2, 2] strides in convolutions are replaced by [1, 1] strides, followed by max-pooling of [1, 2] to maintain the time resolution.
CRNN	Only Conv2D and Conv2D-dw layers until 256 filters are included from Table 1. After this, two B-GRU layers with 80 units each are added.

3. Datasets

In this paper, we evaluate the robustness of the YOHO algorithm on multiple datasets. This section explains the different datasets and how we adapt the YOHO algorithm for each of them.

3.1. Music-Speech Detection

Music-speech detection aims to detect the boundaries of music and speech in audio signals such as radio and TV programs. The neural network performs multi-output detection to allow the simultaneous occurrence of music and speech. The number of output neurons at each time step is six because we are detecting two acoustic classes. We obtained 5 h of audio from the MuSpeak dataset [38]. In addition, we collected 18 h of audio from BBC Radio Devon, which was manually annotated by the authors. Both datasets were roughly split into 50% for training, 30% for validation, and 20% for testing.

There are many openly available datasets with separate files of music and speech, such as MUSAN [39], GTZAN [40,41], Scheirer and Slaney dataset [42], and Instrument Recognition in Musical Audio Signals (IRMAS) [43], to name a few. However, the problem with such datasets is that they are not mixed in the style of TV or radio programmes. Broadcast audio is generally well-mixed with instances of speech over background music, one song fading out and a new song fading in, and so on. In a previous study [17], we presented an approach that artificially synthesises large training sets for music-speech detection. This technique automatically mixes separate files of music and speech in the style of a radio DJ. Various parameters such as audio fade curves and audio ducking are randomised to obtain a variety of synthetic examples. In the current paper, we included 46 h of synthetic examples in the training set. Table 4 shows a brief overview of the contents of each split in the dataset. For a detailed explanation of the training sets and experimental setup, please refer to this study [17].

Table 4. Contents of train, validation, and test datasets for music-speech detection. Real-world radio data was collected from BBC Radio Devon and annotated by the authors. MuSpeak [38] already contains annotations for music and speech. 46 h of artificial radio-like examples were synthesised by the method presented in this study [17].

Dataset Division	Contents
Train	46 h of synthetic radio data + 9 h from BBC Radio Devon + 1 h 30 min from MuSpeak
Validation	5 h from BBC Radio Devon + 2 h from MuSpeak
Test	4 h from BBC Radio Devon + 1 h 42 min from MuSpeak

All the audio files were resampled to 16 kHz. They were converted to mono by averaging the channels before pre-processing. Subsequently, we extracted 64 log-mel bins with a hop size of 10 ms and a window size of 25 ms. The frequencies for the mel spectrogram ranged from 125 Hz to 7.5 kHz. We adopted audio features similar to those used by YamNet [31]. Note that we did not use any regularization such as L2 normalization, spatial dropout, or SpecAugment for this dataset because the training set is large.

We evaluate the model on two different test sets. The first one being our in-house test set, which contains approximately 4.5 h of audio from BBC Radio Devon and MuSpeak [38]. The second one was the MIREX music-speech detection dataset, which contains 27 h of audio from various TV programs.

3.2. TUT Sound Event Detection

The TUT Sound Event Detection dataset focuses on environmental sound detection [28]. It was adopted for the third task of the DCASE challenge 2017. It primarily consists of street recordings with traffic and other activity. Each audio example is 2.56 s. There were six unique audio classes—Brakes Squeaking, Car, Children, Large Vehicle, People Speaking, and People Walking. Thus, to predict the existence of the six classes, plus start and end times, we required 18 output neurons. The more recent DCASE challenges use additional techniques such as semi-supervised learning and source separation, which is not the focus of this study. Hence, we used the dataset from 2017 that contains only strongly labeled data.

The total size of the dataset is approximately 1.5 h. The dataset comes with a four-fold cross-validation setup. The size of this dataset is significantly smaller than the one used for music-speech detection and may not be large enough for our deep learning architecture. Therefore, we applied L2 normalization of 0.001 on the first Conv2D layer. In addition, we included L2 normalization of 0.01 and spatial dropout of 0.1 on all the subsequent Conv2D layers. For data augmentation, we incorporated SpecAugment [36], which randomly drops a sequence of frequency bins or time steps from the input. Note that there were slight differences in our implementation of SpecAugment. We did not use any time warping because it becomes complicated to redefine labels for audio events. In addition, we applied SpecAugment on batches instead of individual examples to save computational time.

The database contained stereo audio files with a sampling rate of 44.1 kHz. These were downmixed to mono before pre-processing. Subsequently, we extracted 40 log-mel bands in the range of 0 to 22,050 Hz. The hop size was 10 ms and the window size was 40 ms. We adopted audio features similar to the baseline system [28] for the task, except that we used a smaller hop size. As the input of the network contains 2.56 s of audio, the input shape is 257×40 corresponding to 257 time steps and 40 mel bins. The output shape of the network is 9×18 , corresponding to 9 time steps and 6 acoustic classes. Note that each time step in this case is 0.284 s, which is different from 0.307 s for music-speech detection. In both cases, we used the same network and the same sequence of convolutional layers. The convolutions with a stride of 2 reduces the temporal dimension by half. Hence, due to different input sizes, the number of time steps is 9 in one case and 26 in the other case. There were no special measures taken to estimate the duration of each time step beforehand. However, in most cases it was somewhere around 0.3 s, due to the hop and window sizes selected for feature extraction.

3.3. Urban-SED

The Urban Sound Event Detection dataset is a purely synthetic dataset generated by using scaper [9]. Each audio example was 10 s. There were ten unique audio classes—Air Conditioner, Car Horn, Children Playing, Dog Bark, Drilling, Engine Idling, Gun Shot, Jackhammer, Siren, and Street Music. The total size of the dataset is about 30 h and contains pre-defined splits for training, validation, and testing. As there were ten audio classes, the number of output neurons in YOHO was 30.

We used the same audio features as explained in Section 3.2. For this dataset, we did not use any SpecAugment because the training set was larger. The L2 normalization and spatial dropout were identical to those used in Section 3.2. As the input of the network contains 10 s of audio, the input shape is 1001×40 corresponding to 1001 time steps and 40 mel bins. The output shape of the network is 32×30 , corresponding to 32 time steps and ten acoustic classes.

4. Results

4.1. Music-Speech Detection

4.1.1. In-House Test Set

Table 5 shows the results on our in-house test set. F-measure was calculated using the sed_eval [37] module with a segment size of 10 ms. We compare the results of YOHO with the CNN and CRNN models explained in Section 2.7. In addition, we compare the performance with CNN and CRNN architectures published in previous research [8,17]. All the deep learning models were trained using the same training set. YOHO obtains the highest F-measure for overall, music, and speech. YOHO significantly outperforms the CNN, which is the segmentation-by-classification version of the model. It is important to note that both models follow the same process for feature extraction and have the same number of parameters. This shows that our regression approach of predicting the acoustic boundaries directly is effective. The other CNN [17] used larger kernel sizes such as 9 and 11, which may have improved the F-measure of Speech.

YOHO also outperforms the three CRNN architectures. CRNN [8] used a kernel size of 7 and CRNN [17] used kernel sizes of 3, 11, and 11. In addition, CRNN [17] used layer normalisation [44] instead of batch normalisation [33]. Therefore, we show that YOHO outperforms a variety of CRNN architectures in the literature.

Table 5. Results on our in-house test set for music-speech detection. The F-measures for overall, music, and speech are presented as percentages. The values in bold indicate the largest number in each column.

Algorithm	F _{overall}	F _{music}	F _{speech}
YOHO	97.22	98.20	94.89
CRNN	96.79	97.84	94.26
CNN	93.89	97.96	85.13
CRNN [17]	96.37	97.37	94.00
CRNN [8]	96.24	97.30	93.80
CNN [17]	95.23	97.72	89.62

4.1.2. MIREX Music-Speech Detection

Table 6 shows the results on the MIREX music-speech detection dataset. YOHO obtains the highest overall F-measure, which makes it the state-of-the-art for music-speech detection. The music F-measure for a CRNN [8] slightly surpassed the YOHO algorithm by 0.1%. However, YOHO obtained the highest F-measure for speech.

Table 6. Evaluation on the MIREX music-speech detection dataset 2018. The results of other studies were obtained from the MIREX website [27]. The F-measures are presented as percentages. The values in bold indicate the largest number in each column.

Algorithm	F _{overall}	F _{music}	F _{speech}
YOHO	90.20	85.66	93.18
CRNN [8]	89.53	85.76	92.21
CRNN [17]	89.09	85.01	92.16
CNN [45]	-	54.78	90.9
Logistic Regression [45]	-	38.99	91.15
ResNet [45]	-	31.24	90.86
MLP [46]	-	49.36	77.18

4.2. TUT Sound Event Detection

Table 7 shows the results on the TUT sound event detection dataset. It also contains the results of the top three performers in the competition. For this competition, they adopted error rate [37] as the main metric. Note that a lower error rate indicates better performance of the algorithm. Furthermore, a segment size of 1 s was adopted to calculate segment-based metrics. The first place in the competition was the CRNN architecture [47]. They used 3×3 kernels followed by B-GRU layers with 32 units. Their model was optimised by a random hyperparameter search [48] for number of layers and units. The second place in the competition adopted a multi-input CNN with 3×3 kernels and a bespoke feature extraction process. The third place adopted a B-GRU model. Note that all three models adopt segmentation-by-classification. YOHO obtained a better error rate than the CRNN [47], CNN [49], and B-GRU [50] models. To ensure that the improvement in performance was not attributed to data augmentation, we re-trained the best CRNN network [47] with SpecAugment. However, it worsened the performance of the algorithm. This may be because the CRNN uses segmentation-by-classification. Therefore, masking series of time steps leads to noise in the labels. However, the YOHO algorithm is relatively robust to this issue as it directly predicts boundaries through regression.

Our results are not state-of-the-art on this dataset. Vesperini et al. [51] adopted a Capsule Neural Network (CapsNet) and binaural short-time Fourier transform (STFT) for feature extraction and obtained an error rate 0.58. Luo et al. [52] presented a Capsule Neural Network Recurrent Neural Network (CapsNet-RNN) that obtained an error rate of 0.57. However, these optimisations were beyond the scope of this study. It is important to note that YOHO is a paradigm and not an architecture. We show that regression outperforms segmentation-by-classification for multiple models. Future research can explore how YOHO can be optimised by adopting a CapsNet-style architecture.

Table 7. Results on the TUT sound event detection dataset. The value in bold indicates the algorithm with the lowest error rate.

Algorithm	Error Rate
CapsNet-RNN [52]	0.57
CapsNet [51]	0.59
YOHO	0.75
CRNN [47]	0.79
CNN [49]	0.81
B-GRU [50]	0.83

4.3. Urban-SED

Table 8 shows the results on the Urban-SED dataset for overall F-measure. A comparison of class-wise performance is also presented in Figure 3. The YOHO algorithm is compared with the CRNN and CNN model presented by Salamon et al. [9]. YOHO obtains the highest overall F-measure. Among class-wise F-measures, YOHO obtains the highest for Children Playing, Dog Bark, Drilling, Gun Shot, Siren, and Street Music. CRNN obtains the highest for Air Conditioner and Engine Idling. CNN obtains the highest for Car Horn and Jackhammer.

As you can see in Table 8, Martín-Morató et al. [53] adopted sound event envelope estimation on a CRNN model to improve the overall F-measure to 64.7%, compared to 59.5% obtained by YOHO. In future research, YOHO's performance can be improved by incorporating techniques like envelope estimation. In addition, weakly supervised sound event detection with envelope estimation has further improved the performance of the CRNN on this dataset [54].

Table 8. Segment-based overall F-measure on the Urban-SED dataset. The value in bold indicates the algorithm with the highest F-measure.

Algorithm	F _{overall}
CRNN with envelope estimation [53]	64.70
YOHO	59.50
CNN [9]	56.88
CRNN [9]	55.96

4.4. Speed of Prediction

In this section, we compare the inference times of YOHO, CNN and CRNN models for music-speech detection. This experiment was performed on the in-house test set explained in Section 3.1. To calculate the inference time, the prediction was made over the entire test set. Later, the inference time was divided by the number of hours of audio to obtain the average time taken per hour of audio. As we are adopting Google Colab for experiments, we ensured that all models were tested within the same runtime session. This way, we ensure that the same computing resources were given to YOHO, CNN, and CRNN. While training the models in earlier runtime sessions, we had stored their weights on Google Drive. When running the experiment to calculate inference times, these weights were loaded from Google Drive. Note that separate runtime sessions were used to calculate inference times over CPU and GPU as shown in Figure 4, however the same session was used for inter-model comparison. Some important aspects of the system configuration were—Intel Xeon CPU processor, 12 GB RAM, and Tesla P100 GPU (only when GPU was used).

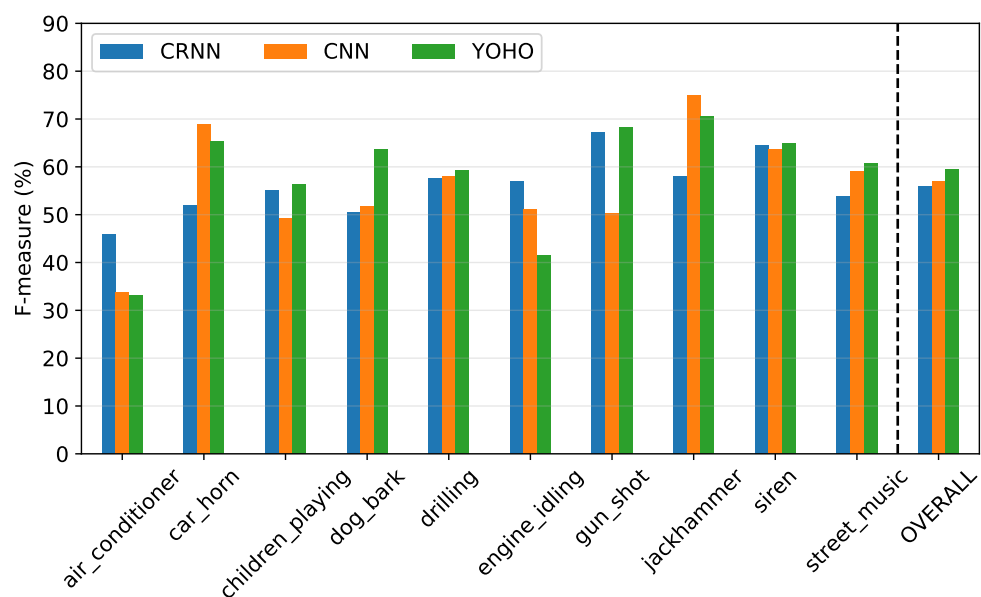
**Figure 3.** Segment-based F-measures for each class on the Urban-SED dataset calculated using segment-size of 1 s.

Figure 4 compares the inference times of YOHO, CNN and CRNN models for music-speech detection on the in-house test set. The CNN and CRNN models were explained in Section 2.7. YOHO and the CNN had exactly the same number of parameters, which is 3.9 million. The only difference is that the CNN adopts frame-based classification instead of regression. The CRNN model had 1.3 million parameters, which was less than the CNN and YOHO. On the CPU, the prediction time of YOHO was 14 times faster than the CNN and 5

times faster than the CRNN. On the Graphical Processing Unit (GPU), the prediction time of YOHO was 6 times faster than the CNN and 4 times faster than the CRNN. The increase in prediction speed is because YOHO has to predict only 26×6 neurons, whereas the CNN and CRNN have to predict 801×2 neurons. Despite the CRNN having fewer parameters, YOHO is significantly faster.

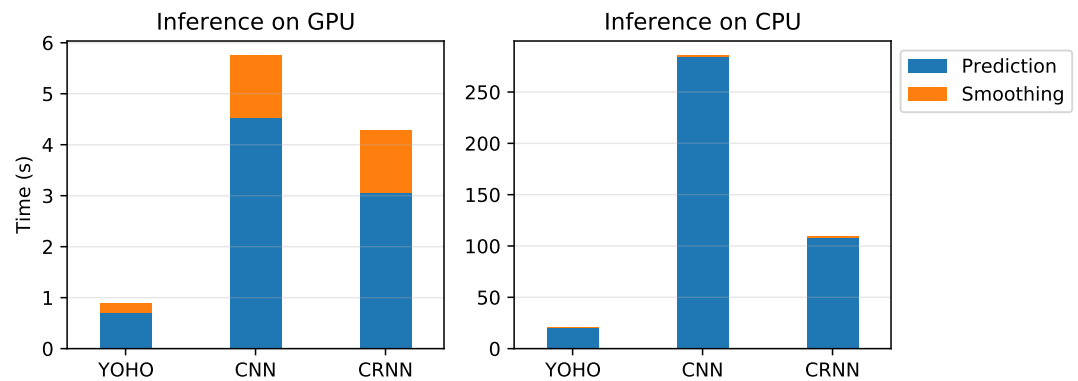


Figure 4. Average time taken to make predictions on 1 h of audio for music-speech detection. ‘Prediction’ refers to the time taken by the network to make predictions. ‘Smoothing’ is the post-processing step to parse the output of the network. The GPU used for inference was the Tesla P100.

As YOHO outputs acoustic boundaries directly, the post-processing and smoothing for YOHO was 7 times faster than the CNN and CRNN. Note that the smoothing is performed only on the CPU.

5. Discussion

The results in Section 4 show that YOHO has multiple advantages over the state-of-the-art CRNN architecture. We examined the model for two different tasks—music-speech detection and environmental sound event detection. Music-speech detection is relatively a simpler task because of a larger and diverse training set. Additionally, there are only two acoustic classes to predict. On the other hand, environmental sound event detection was harder because of smaller and lower quality training sets. In addition, the number of acoustic classes was greater. However, in both scenarios, YOHO generalised better than CRNN and CNN. YOHO obtained state-of-the-art performance for music-speech detection on the MIREX 2018 competition dataset. We understand that YOHO has not obtained state-of-the-art performance on TUT Sound Events and Urban-SED datasets. However, it is important to note that the purpose of this study is to shift the paradigm from frame-based classification to regression for audio segmentation and sound event detection. There is a vast body of research involving CNN and CRNN architectures. It is not within the capacity of this study to incorporate all these optimisations for YOHO. As this is the first study that explores this paradigm, we believe that optimisations such as weak label learning [11] and envelope estimation [53] will improve YOHO’s performance.

We also explored the idea of creating a regression-based CRNN that adopts the YOHO paradigm. We replaced the Conv1D layer with a B-GRU block. However, this slightly worsened the performance of the algorithm. This is because the YOHO network has many convolutional layers that reduces the temporal resolution from 801 to 26. Hence, the B-GRU blocks may not be effective on such a small number of time steps. However, alternative structures such as CNN-transformers [55] may be a promising avenue to explore.

YOHO was significantly quicker than the CNN and CRNN models because it had to predict fewer outputs and computationally cheaper post-processing. As explained in the paper, the output produced by YOHO is more end-to-end. For example, the output dimensions in music-speech detection is 26×6 for YOHO versus 801×2 for the CRNN. This corresponds

to 156 output neurons for YOHO and 1602 for the CRNN. Furthermore, the CRNN needs to convert frame-based classifications to time boundaries. However, YOHO directly outputs the time boundaries. Due to the above reasons, YOHO is significantly quicker. Due to faster inference, YOHO is more suitable for real-time applications such as surveillance, self-driving automobiles, bioacoustic monitoring, and real-time remixing.

6. Conclusions

In this paper, we proposed a novel paradigm called YOHO for audio segmentation and sound event detection. It obtained state-of-the-art performance for music-speech detection and surpassed the CRNN and CNN's performance for environmental audio. YOHO presents sound event detection differently from the traditional segmentation-by-classification approach. We primarily adapted the MobileNet architecture [30] to develop the YOHO paradigm. Future developments in the network architecture for YOHO would lead to improvements in performance. For instance, adding skip connections through ResNets [56] or by including Inception blocks [57]. Furthermore, there is scope to create hybrid architectures such as CNN-transformers [55] by adopting the YOHO paradigm.

Although YOHO's output is more end-to-end by predicting acoustic boundaries directly, it is limited by the time-resolution of the input, which is the mel spectrogram. It would be interesting to explore YOHO with raw audio, which would make the sound event detection pipeline completely end-to-end. Moreover, the YOHO approach is relevant to related tasks such as singing voice detection. Furthermore, recent studies have successfully combined sound event detection with source separation and semi-supervised learning [10,58]. Future work could explore how YOHO would perform in these scenarios.

Author Contributions: Conceptualization, S.V. and D.M.; methodology, S.V. and D.M.; software, S.V.; investigation, S.V.; writing—original draft preparation, S.V.; writing—review and editing, D.M. and S.V.; supervision, D.M. and E.R.M.; project administration, E.R.M.; funding acquisition, E.R.M. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by Engineering and Physical Sciences Research Council (EPSRC) grant EP/S026991/1.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data provided by BBC Radio Devon is copyrighted material and cannot be shared. Synthetic radio data for music-speech detection can be generated by using techniques in this paper [17]. MuSpeak [38] is an openly available annotated dataset for music-speech detection, which can be utilised by researchers for validation and testing. TUT Sound Events 2017 and Urban-SED datasets are publicly available. The code associated with this paper is openly available in this GitHub repository (<https://github.com/satvik-venkatesh/you-only-hear-once/>, accessed on 2 March 2022).

Acknowledgments: The authors would like to thank Blai Meléndez Catalán for helping us evaluate our model on the MIREX music-speech competition dataset. We thank Justin Salamon for providing insights on the Urban-SED dataset [9] and details of the results presented in Figure 3. Research in this paper was conducted on Google Colab and we are thankful for their service.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

B-GRU	Bidirectional Gated Recurrent Unit
B-LSTM	Bidirectional Long Short-Term Memory
BBC	British Broadcasting Corporation
CapsNet	Capsule Neural Network
CapsNet-RNN	Capsule Neural Network Recurrent Neural Network
CNN	Convolutional Neural Network
CRNN	Convolutional Recurrent Neural Network
DCASE	Detection and Classification of Acoustic Scenes and Events
GPU	Graphical Processing Unit
MIREX	Music Information Retrieval Evaluation eXchange
MLP	Multi-Layer Perceptron
RAM	Random Access Memory
STFT	Short-Time Fourier Transform
Urban-SED	Urban Sound Event Detection
YOHO	You Only Hear Once
YOLO	You Only Look Once

References

1. Butko, T.; Nadeu, C. Audio segmentation of broadcast news in the Albayzin-2010 evaluation: Overview, results, and discussion. *EURASIP J. Audio Speech Music Process.* **2011**, *2011*, 1. [CrossRef]
2. Elizalde, B.; Raja, B.; Vincent, E. Task 4: Large-Scale Weakly Supervised Sound Event Detection for Smart Cars. 2017. Available online: <http://dcase.community/challenge2017/task-large-scale-sound-event-detection> (accessed on 2 March 2022).
3. Radhakrishnan, R.; Divakaran, A.; Smaragdakis, A. Audio analysis for surveillance applications. In Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, NY, USA, 16–19 October 2005; pp. 158–161.
4. Salamon, J.; Bello, J.P.; Farnsworth, A.; Robbins, M.; Keen, S.; Klinck, H.; Kelling, S. Towards the automatic classification of avian flight calls for bioacoustic monitoring. *PLoS ONE* **2016**, *11*, e0166866.
5. Ramirez, M.M.; Stoller, D.; Moffat, D. A Deep Learning Approach to Intelligent Drum Mixing with the Wave-U-Net. *J. Audio Eng. Soc.* **2021**, *69*, 142–151. [CrossRef]

6. Theodorou, T.; Mporas, I.; Fakotakis, N. An overview of automatic audio segmentation. *Int. J. Inf. Technol. Comput. Sci. (IJITCS)* **2014**, *6*, 1. [CrossRef]
7. Huang, R.; Hansen, J.H. Advances in unsupervised audio classification and segmentation for the broadcast news and NGSW corpora. *IEEE Trans. Audio Speech Lang. Process.* **2006**, *14*, 907–919. [CrossRef]
8. Venkatesh, S.; Moffat, D.; Kirke, A.; Shakeri, G.; Brewster, S.; Fachner, J.; Odell-Miller, H.; Street, A.; Farina, N.; Banerjee, S.; et al. Artificially Synthesising Data for Audio Classification and Segmentation to Improve Speech and Music Detection in Radio Broadcast. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021; pp. 636–640.
9. Salamon, J.; MacConnell, D.; Cartwright, M.; Li, P.; Bello, J.P. Scaper: A library for soundscape synthesis and augmentation. In Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), New Paltz, NY, USA, 15–18 October 2017; pp. 344–348.
10. Turpault, N.; Serizel, R.; Shah, A.; Salamon, J. Sound Event Detection in Domestic Environments with Weakly Labeled Data and Soundscape Synthesis. In Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE), New York, NY, USA, 25–26 October 2019; p. 253.
11. Miyazaki, K.; Komatsu, T.; Hayashi, T.; Watanabe, S.; Toda, T.; Takeda, K. Weakly-supervised sound event detection with self-attention. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 66–70.
12. Gemmeke, J.F.; Ellis, D.P.; Freedman, D.; Jansen, A.; Lawrence, W.; Moore, R.C.; Plakal, M.; Ritter, M. Audio set: An ontology and human-labeled dataset for audio events. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017; pp. 776–780.
13. Hershey, S.; Ellis, D.P.; Fonseca, E.; Jansen, A.; Liu, C.; Moore, R.C.; Plakal, M. The benefit of temporally-strong labels in audio event classification. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021; pp. 366–370.
14. Gimeno, P.; Viñals, I.; Ortega, A.; Miguel, A.; Lleida, E. Multiclass audio segmentation based on recurrent neural networks for broadcast domain data. *EURASIP J. Audio Speech Music Process.* **2020**, *2020*, 1–19. [CrossRef]
15. Lemaire, Q.; Holzapfel, A. Temporal Convolutional Networks for Speech and Music Detection in Radio Broadcast. In Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR), Delft, The Netherlands, 4–8 November 2019.
16. Cakır, E.; Parascandolo, G.; Heittola, T.; Huttunen, H.; Virtanen, T. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2017**, *25*, 1291–1303. [CrossRef]
17. Venkatesh, S.; Moffat, D.; Miranda, E.R. Investigating the Effects of Training Set Synthesis for Audio Segmentation of Radio Broadcast. *Electronics* **2021**, *10*, 827. [CrossRef]
18. Dieleman, S.; Schrauwen, B. End-to-end learning for music audio. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 4–9 May 2014; pp. 6964–6968.
19. Lee, J.; Park, J.; Kim, T.; Nam, J. Raw Waveform-based Audio Classification Using Sample-level CNN Architectures. In Proceedings of the Machine Learning for Audio Signal Processing Workshop, Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 4–9 December 2017.
20. Phan, H.; Maaß, M.; Mazur, R.; Mertins, A. Random regression forests for acoustic event detection and classification. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2014**, *23*, 20–31. [CrossRef]
21. Xu, Y.; Du, J.; Dai, L.R.; Lee, C.H. A regression approach to speech enhancement based on deep neural networks. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2014**, *23*, 7–19. [CrossRef]
22. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 779–788.
23. Zsebök, S.; Nagy-Egri, M.F.; Barnaföldi, G.G.; Laczi, M.; Nagy, G.; Vaskuti, É.; Garamszegi, L.Z. Automatic bird song and syllable segmentation with an open-source deep-learning object detection method—a case study in the Collared Flycatcher. *Ornis Hung.* **2019**, *27*, 59–66. [CrossRef]
24. Segal, Y.; Fuchs, T.S.; Keshet, J. SpeechYOLO: Detection and Localization of Speech Objects. *arXiv* **2019**, arXiv:1904.07704.
25. Algabri, M.; Mathkour, H.; Bencherif, M.A.; Alsulaiman, M.; Mekhtiche, M.A. Towards deep object detection techniques for phoneme recognition. *IEEE Access* **2020**, *8*, 54663–54680. [CrossRef]
26. Zhou, X.; Wang, D.; Krähenbühl, P. Objects as points. *arXiv* **2019**, arXiv:1904.07850.
27. Schlüter, J.; Doukhan, D.; Meléndez-Catalán, B. MIREX Challenge: Music and/or Speech Detection. 2018. Available online: https://www.music-ir.org/mirex/wiki/2018:Music_and/or_Speech_Detection (accessed on 2 March 2022).
28. Mesaros, A.; Heittola, T.; Diment, A.; Elizalde, B.; Shah, A.; Vincent, E.; Raj, B.; Virtanen, T. DCASE 2017 challenge setup: Tasks, datasets and baseline system. In Proceedings of the Workshop on Detection and Classification of Acoustic Scenes and Events, Munich, Germany, 16 November 2017.

29. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
30. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
31. Plakal, M.; Ellis, D. YAMNet. 2020. Available online: <https://github.com/tensorflow/models/tree/master/research/audioset/yamnet/> (accessed on 2 March 2022).
32. Sifre, L. Rigid-Motion Scattering for Image Classification. Ph.D. Thesis, Ecole Normale Supérieure, Paris, France, 2014.
33. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning (ICLR), San Diego, CA, USA, 7–9 May 2015; pp. 448–456.
34. Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. In Proceedings of the 2nd International Conference on Learning Representations (ICLR), Banff, AB, Canada, 14–16 April 2014.
35. Yao, Y.; Rosasco, L.; Caponnetto, A. On early stopping in gradient descent learning. *Constr. Approx.* **2007**, *26*, 289–315. [CrossRef]
36. Park, D.S.; Chan, W.; Zhang, Y.; Chiu, C.C.; Zoph, B.; Cubuk, E.D.; Le, Q.V. SpecAugment: A simple data augmentation method for automatic speech recognition. In Proceedings of the Interspeech, Graz, Austria, 15–19 September 2019; pp. 2613–2617.
37. Mesaros, A.; Heittola, T.; Virtanen, T. Metrics for polyphonic sound event detection. *Appl. Sci.* **2016**, *6*, 162. [CrossRef]
38. MuSpeak Team. MIREX MuSpeak Sample Dataset. 2015. Available online: <http://mirg.city.ac.uk/datasets/muspeak/> (accessed on 2 March 2022).
39. Snyder, D.; Chen, G.; Povey, D. Musan: A music, speech, and noise corpus. *arXiv* **2015**, arXiv:1510.08484.
40. Tzanetakis, G.; Cook, P. Marsyas: A framework for audio analysis. *Organised Sound* **2000**, *4*, 169–175. [CrossRef]
41. Tzanetakis, G.; Cook, P. Musical genre classification of audio signals. *IEEE Trans. Speech Audio Process.* **2002**, *10*, 293–302. [CrossRef]
42. Scheirer, E.; Slaney, M. Construction and evaluation of a robust multifeature speech/music discriminator. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Munich, Germany, 21–24 April 1997; Volume 2, pp. 1331–1334.
43. Bosch, J.J.; Janer, J.; Fuhrmann, F.; Herrera, P. A Comparison of Sound Segregation Techniques for Predominant Instrument Recognition in Musical Audio Signals. In Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR), Porto, Portugal, 8–12 October 2012; pp. 559–564.
44. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer normalization. *arXiv* **2016**, arXiv:1607.06450.
45. Marolt, M. Music/Speech Classification and Detection Submission for MIREX 2018. Music Inf. Retr. Eval. eXchange MIREX. 2018. Available online: <https://www.music-ir.org/mirex/abstracts/2018/MM2.pdf> (accessed on 2 March 2022).
46. Choi, M.; Lee, J.; Nam, J. Hybrid Features for Music and Speech Detection. Music Inf. Retr. Eval. eXchange (MIREX). 2018. Available online: <https://www.music-ir.org/mirex/abstracts/2018/LN1.pdf> (accessed on 2 March 2022).
47. Adavanne, S.; Virtanen, T. A Report on Sound Event Detection with Different Binaural Features. In Proceedings of the Detection and Classification of Acoustic Scenes and Events (DCASE), Munich, Germany, 16 November 2017.
48. Bergstra, J.; Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
49. Jeong, I.Y.; Lee, S.; Han, Y.; Lee, K. Audio Event Detection Using Multiple-Input Convolutional Neural Network. In Proceedings of the Detection and Classification of Acoustic Scenes and Events (DCASE), Munich, Germany, 16 November 2017.
50. Lu, R.; Duan, Z. Bidirectional GRU for Sound Event Detection. In Proceedings of the Detection and Classification of Acoustic Scenes and Events (DCASE), Munich, Germany, 16 November 2017.
51. Vesperini, F.; Gabrielli, L.; Principi, E.; Squartini, S. Polyphonic sound event detection by using capsule neural networks. *IEEE J. Sel. Top. Signal Process.* **2019**, *13*, 310–322. [CrossRef]
52. Luo, L.; Zhang, L.; Wang, M.; Liu, Z.; Liu, X.; He, R.; Jin, Y. A System for the Detection of Polyphonic Sound on a University Campus Based on CapsNet-RNN. *IEEE Access* **2021**, *9*, 147900–147913. [CrossRef]
53. Martín-Morató, I.; Mesaros, A.; Heittola, T.; Virtanen, T.; Cobos, M.; Ferri, F.J. Sound event envelope estimation in polyphonic mixtures. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 935–939.
54. Dinkel, H.; Wu, M.; Yu, K. Towards duration robust weakly supervised sound event detection. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2021**, *29*, 887–900. [CrossRef]
55. Kong, Q.; Xu, Y.; Wang, W.; Plumbley, M.D. Sound event detection of weakly labelled data with CNN-transformer and automatic threshold optimization. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2020**, *28*, 2450–2460. [CrossRef]
56. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
57. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9.

-
58. Turpault, N.; Serizel, R.; Wisdom, S.; Erdogan, H.; Hershey, J.R.; Fonseca, E.; Seetharaman, P.; Salamon, J. Sound Event Detection and Separation: A Benchmark on Desed Synthetic Soundscapes. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021; pp. 840–844.