Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection

Emre Çakır, Giambattista Parascandolo, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen

***Abstract*—Sound events often occur in unstructured environments where they exhibit wide variations in their frequency content and temporal structure. Convolutional neural networks (CNNs) are able to extract higher level features that are invariant to local spectral and temporal variations. Recurrent neural networks (RNNs) are powerful in learning the longer term temporal context in the audio signals. CNNs and RNNs as classifiers have recently shown improved performances over established methods in various sound recognition tasks. We combine these two approaches in a convolutional recurrent neural network (CRNN) and apply it on a polyphonic sound event detection task. We compare the performance of the proposed CRNN method with CNN, RNN, and other established methods, and observe a considerable improvement for four different datasets consisting of everyday sound events.**

***Index Terms*—Convolutional neural networks (CNNs), deep neural networks, recurrent neural networks (RNNs), sound event detection.**

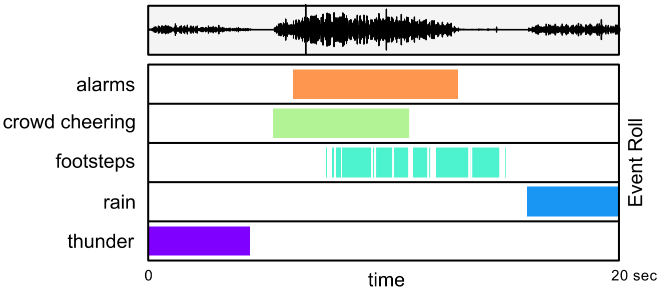

Fig. 1. Sound events in a polyphonic recording synthesized with isolated sound event samples. Upper panel: audio waveform, lower panel: sound event class activity annotations.

I. Introduction

In our daily lives, we encounter a rich variety of sound events such as dog bark, footsteps, glass smash and thunder. Sound event detection (SED), or acoustic event detection, deals with the automatic identification of these sound events. The aim of SED is to detect the onset and offset times for each sound event in an audio recording and associate a textual descriptor, i.e., a label for each of these events. SED has been drawing a surging amount of interest in recent years with applications including audio surveillance [1], healthcare monitoring [2], urban sound analysis [3], multimedia event detection [4] and bird call detection [5].

In the literature the terminology varies between authors; common terms being sound event *detection*, *recognition*, *tagging* and *classification*. Sound events are defined with pre-determined labels called sound event *classes*. In our work, sound event classification, sound event recognition, or sound event tagging, all refer to labeling an audio recording with the sound event classes present, regardless of the onset/offset times. On the other hand, an SED task includes both onset/offset detection for the classes present in the recording and classification within the estimated onset/offset, which is typically the requirement in a real-life scenario.

Sound events often occur in unstructured environments in real-life. Factors such as environmental noise and overlapping sources are present in the unstructured environments and they may introduce a high degree of variation among the sound events from the same sound event class [6]. Moreover, there can be multiple sound sources that produce sound events belonging to the same class, e.g., a dog bark sound event can be produced from several breeds of dogs with different acoustic characteristics. These factors mainly represent the challenges over SED in real-life situations.

SED where at most one simultaneous sound event is detected at a given time instance is called *monophonic* SED. Monophonic SED systems can only detect at most one sound event for any time instance regardless of the number of sound events present. If the aim of the system is to detect all the events happening at a time, this is a drawback concerning the real-life applicability of such systems, because in such a scenario, multiple sound events are very likely to overlap in time. For instance, an audio recording from a busy street may contain footsteps, speech and car horn, all appearing as a mixture of events. An illustration of a similar situation is given in Fig. 1, where as many as three different sound events appear at the same time in a mixture. A more suitable method for such a real-life scenario is *polyphonic* SED, where multiple overlapping sound events can be detected at any given time instance.

Manuscript received July 13, 2016; revised November 2, 2016 and January 20, 2017; accepted February 15, 2017. Date of current version May 23, 2017. The work was supported by the European Research Council under the European Unions H2020 Framework Programme through ERC Grant Agreement 637422 EVERYSOUND. The work of G. Parascandolo was supported by Google's Faculty Research award. The guest editor coordinating the review of this manuscript and approving it for publication was Dr. J. P. Bello. *(G. Parascandolo and E. Cakir contributed equally to this work.) (Corresponding author: Emre Cakir.)*

The authors are with the Department of Signal Processing, Tampere University of Technology, Tampere 33101, Finland (e-mail: emre.cakir@tut.fi; giambattista.parascandolo@tut.fi; toni.heittola@tut.fi; heikki.huttunen@tut.fi; tuomas.virtanen@tut.fi).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASLP.2017.2690575

SED can be approached either as *scene-dependent* or *scene-independent*. In the former, the information about the acoustic scene is provided to the system both at training and test time, and a different model can therefore be trained for each scene. In the latter, there is no information about the acoustic scene given to the system.

Previous work on sound events has been mostly focused on sound event classification, where audio clips consisting of sound events are classified. Apart from established classifiers—such as support vector machines [1], [3]—deep learning methods such as deep belief networks [7], convolutional neural networks (CNN) [8]–[10] and recurrent neural networks (RNN) [4], [11] have been recently proposed. Initially, the interest on SED was more focused on monophonic SED. Gaussian mixture model (GMM) - Hidden Markov model (HMM) based modeling—an established method that has been widely used in automatic speech recognition—has been proposed to model individual sound events with Gaussian mixtures and detect each event through HMM states using Viterbi algorithm [12], [13]. With the emergence of more advanced deep learning techniques and publicly available real-life databases that are suitable for the task, polyphonic SED has attracted more interest in recent years. Non-negative matrix factorization (NMF) based source separation [14] and deep learning based methods (such as feedforward neural networks (FNN) [15], CNN [16] and RNN [11]) have been shown to perform significantly better compared to established methods such as GMM-HMM for polyphonic SED.

Deep neural networks [17] have recently achieved remarkable success in several domains such as image recognition [18], [19], speech recognition [20], [21], machine translation [22], even integrating multiple data modalities such as image and text in image captioning [23]. In most of these domains, deep learning represents the state-of-the-art.

Feedforward neural networks have been used in monophonic [7] and polyphonic SED in real-life environments [15] by processing concatenated input frames from a small time window of the spectrogram. This simple architecture—while vastly improving over established approaches such as GMM-HMMs [24] and NMF source separation based SED [25], [26]—presents two major shortcomings: (1) it lacks both time and frequency invariance—due to the fixed connections between the input and the hidden units—which would allow to model small variations in the events; (2) temporal context is restricted to short time windows, preventing effective modeling of typically longer events (*e.g.*, rain) and events correlations.

CNNs [27] can address the former limitation by learning filters that are shifted in both time and frequency [8], lacking however longer temporal context information. Recurrent neural networks (RNNs), which have been successfully applied to automatic speech recognition (ASR) [20] and polyphonic SED [11], solve the latter shortcoming by integrating information from the earlier time windows, presenting a theoretically unlimited context information. However, RNNs do not easily capture the invariance in the frequency domain, rendering a high-level modeling of the data more difficult. In order to benefit from both approaches, the two architectures can be combined into a single network with convolutional layers followed by recurrent layers, often referred to as convolutional recurrent neural network (CRNN). Similar approaches combining CNNs and RNNs have been presented recently in ASR [21], [28], [29] and music classification [30].

In this paper we propose the use of multi-label convolutional recurrent neural network for polyphonic, scene-independent sound event detection in real-life recordings. This approach integrates the strengths of both CNNs and RNNs, which have shown excellent performance in acoustic pattern recognition applications [4], [8]–[10], while overcoming their individual weaknesses. We evaluate the proposed method on three datasets of real-life recordings and compare its performance to FNN, CNN, RNN and GMM baselines. The proposed method is shown to outperform previous sound event detection approaches.

The rest of the paper is organized as follows. In Section II the problem of polyphonic SED in real-life environments is described formally and the CRNN architecture proposed for the task is presented. In Section III we present the evaluation framework used to measure the performance of the different neural networks architectures. In Section IV experimental results, discussions over the results and comparisons with baseline methods are reported. In Section V we summarize our conclusions from this work.

II. Method

A. Problem Formulation

The aim of polyphonic SED is to temporally locate and label the sound event classes present in a polyphonic audio signal. Polyphonic SED can be formulated in two stages: sound representation and classification. In sound representation stage, frame-level sound features (such as mel band energies and mel frequency cepstral coefficients (MFCC)) are extracted for each time frame t in the audio signal to obtain a feature vector $\mathbf{x}_t \in \mathbb{R}^F$, where $F \in \mathbb{N}$ is the number of features per frame. In the classification stage, the task is to estimate the probabilities $p(\mathbf{y}_t(k) \mid \mathbf{x}_t, \boldsymbol{\theta})$ for event classes $k = 1, 2, \ldots, K$ in frame t, where $\boldsymbol{\theta}$ denotes the parameters of the classifier. The event activity probabilities are then binarized by thresholding, *e.g.* over a constant, to obtain event activity predictions $\hat{\mathbf{y}}_t \in \mathbb{R}^K$.

The classifier parameters $\boldsymbol{\theta}$ are trained by supervised learning, and the target outputs $\mathbf{y}_t$ for each frame are obtained from the onset/offset annotations of the sound event classes. If class k is present during frame t, $\mathbf{y}_t(k)$ will be set to 1, and 0 otherwise. The trained model will then be used to predict the activity of the sound event classes when the onset/offset annotations are unavailable, as in real-life situations.

For polyphonic SED, the target binary output vector $\mathbf{y}_t$ can have multiple non-zero elements since several classes can be present in the same frame t. Therefore, polyphonic SED can be formulated as a multi-label classification problem in which the sound event classes are located by multi-label classification over consecutive time frames. By combining the classification results over consecutive time frames, the onset/offset times for each class can be determined.

Sound events possess temporal characteristics that can be beneficial for SED. Certain sound events can be easily distinguished by their impulsive characteristics (*e.g.*, glass smash), while some sound events typically continue for a long time period (*e.g.*, rain). Therefore, classification methods that can preserve the temporal context along the sequential feature vectors are very suitable for SED. For these methods, the input features are presented as a context window matrix $\mathbf{X}_{t:t+T-1}$, where $T \in \mathbb{N}$ is the number of frames that defines the sequence length of the temporal context, and the target output matrix $\mathbf{Y}_{t:t+T-1}$ is composed of the target outputs $\mathbf{y}_t$ from frames t to $t+T-1$. For the sake of simplicity and ease of notation, $\mathbf{X}$ will be used to denote $\mathbf{X}_{t:t+T-1}$—and similarly $\mathbf{Y}$ for $\mathbf{Y}_{t:t+T-1}$—throughout the rest of the paper.

B. Proposed Method

The CRNN proposed in this work, depicted in Fig. 2, consists of four parts: (1) at the top of the architecture, a time-frequency representation of the data (a context window of F log mel band energies over T frames) is fed to $L_c \in \mathbb{N}$ convolutional layers with non-overlapping pooling over frequency axis; (2) the feature maps of the last convolutional layer are stacked over the frequency axis and fed to $L_r \in \mathbb{N}$ recurrent layers; (3) a single feedforward layer with sigmoid activation reads the final recurrent layer outputs and estimates event activity probabilities for each frame and (4) event activity probabilities are binarized by thresholding over a constant to obtain event activity predictions.

In this structure the convolutional layers act as feature extractors, the recurrent layers integrate the extracted features over time thus providing the context information, and finally the feedforward layer produce the activity probabilities for each class. The stack of convolutional, recurrent and feedforward layers is trained jointly through backpropagation. Next, we present the general network architecture in detail for each of the four parts in the proposed method.

1) Convolutional layers: Context window of log mel band energies $\mathbf{X} \in \mathbb{R}^{F \times T}$ is fed as input to the CNN layers with two-dimensional convolutional filters. For each CNN layer, after passing the feature map outputs through an activation function (rectified linear unit (ReLU) used in this work), non-overlapping max pooling is used to reduce the dimensionality of the data and to provide more frequency invariance. As depicted in Fig. 2, the time dimension is maintained intact (i.e., does not shrink) by computing the max pooling operation in the frequency dimension only—as done in [21], [31]—and by zero-padding the inputs to the convolutional layers (also known as *same* convolution). This is done in order to preserve alignment between each target output vector $\mathbf{y}_t$ and hidden activations $\mathbf{h}_t$.

After L_c convolutional layers, the output of the CNN is a tensor $\mathcal{H} \in \mathbb{R}^{M \times F' \times T}$, where M is the number of feature maps for the last CNN layer, and F' is the number of frequency bands remaining after several pooling operations through CNN layers.

2) Recurrent layers: After stacking the feature map outputs over the frequency axis, the CNN output $\mathbf{H} \in \mathbb{R}^{(M \cdot F') \times T}$ for layer L_c is fed to the RNN as a sequence of frames $\mathbf{h}_t^{L_c}$. The RNN part consists of L_r stacked recurrent layers each computing and outputting a hidden vector $\mathbf{h}_t$ for each frame as

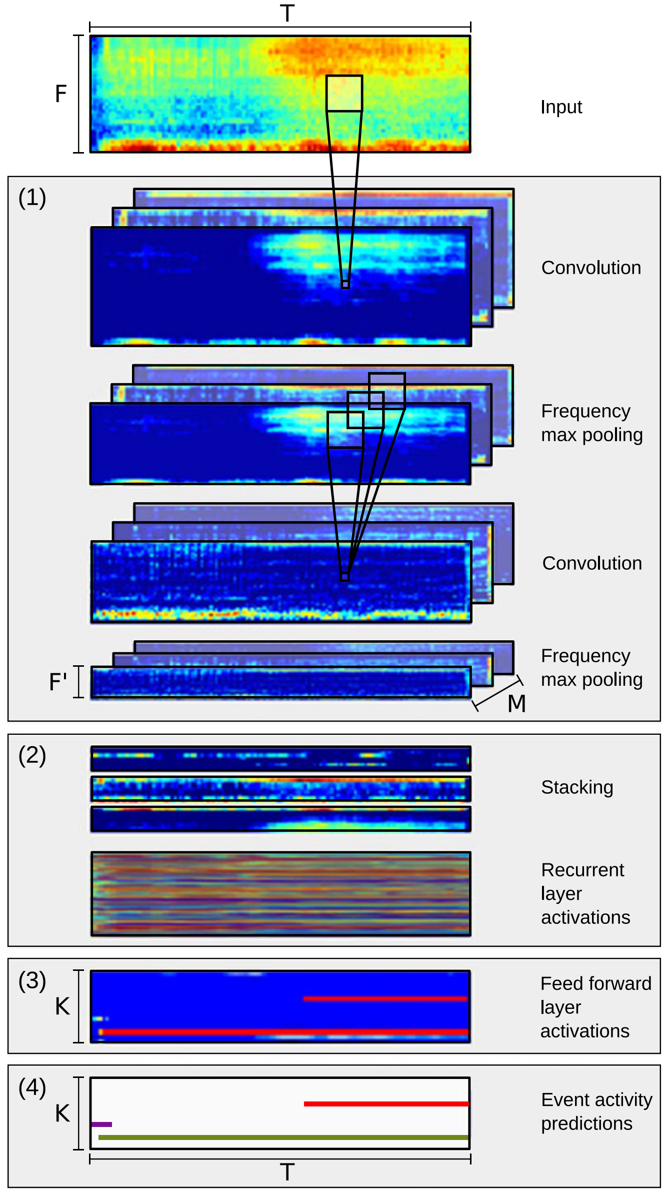

Fig. 2. Overview of the proposed CRNN method. (1): Multiple convolutional layers with max pooling in frequency axis, (2): The outputs of the last convolutional layer stacked over frequency axis and fed to multiple stacked recurrent layers, (3): feedforward layer as output layer and (4): binarization of event activity probabilities.

$$\begin{aligned}
\mathbf{h}_t^{L_c+1} &= \mathcal{F}(\mathbf{h}_t^{L_c}, \mathbf{h}_{t-1}^{L_c+1}) \\
\mathbf{h}_t^{L_c+2} &= \mathcal{F}(\mathbf{h}_t^{L_c+1}, \mathbf{h}_{t-1}^{L_c+2}) \\
&\vdots \\
\mathbf{h}_t^{L_c+L_r} &= \mathcal{F}(\mathbf{h}_t^{L_c+L_r-1}, \mathbf{h}_{t-1}^{L_c+L_r}) \qquad (1)
\end{aligned}$$

The function $\mathcal{F}$, which can represent a long short term memory (LSTM) unit [32] or gated recurrent unit (GRU) [33], has two inputs: The output of the current frame of the previous layer (*e.g.,* $\mathbf{h}_t^{L_c}$), and the output of the previous frame of the current layer (*e.g.,* $\mathbf{h}_{t-1}^{L_c+1}$).

3) Feedforward layer: recurrent layers are followed by a single feedforward layer which will be used as the output layer of the network. The feedforward layer outputs are obtained from the last recurrent layer activations $\mathbf{h}_t^{L_c+L_r}$ as

$$\mathbf{h}_t^{L_c+L_r+1} = \mathcal{G}(\mathbf{h}_t^{L_c+L_r}), \tag{2}$$

where $\mathcal{G}$ represents a feedforward layer with sigmoid activation. Feedforward layer applies the same set of weights for the features extracted from each frame.

4) Binarization: The outputs $\mathbf{h}_t^{L_c+L_r+1}$ of the feedforward layer are used as the event activity probabilities for each class $k = 1, 2, ...K$ as

$$p(\mathbf{y}_t(k) \mid \mathbf{x}_{0:t}, \boldsymbol{\theta}) = \mathbf{h}_t^{L_c+L_r+1} \tag{3}$$

where K is the number of classes and $\boldsymbol{\theta}$ represents the parameters of all the layers of the network combined. Finally, event activity predictions $\hat{\mathbf{y}}_t$ are obtained by thresholding the probabilities over a constant $C \in (0, 1)$ as

$$\hat{\mathbf{y}}_t(k) = \begin{cases} 1, & p(\mathbf{y}_t(k) \mid \mathbf{x}_{0:t}, \boldsymbol{\theta}) \geq C \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Regularization: In order to reduce overfitting, we experimented with dropout [34] regularization in the network, which has proven to be extremely effective in several deep learning applications [18]. The basic idea behind dropout is to temporarily remove at training time a certain portion of hidden units from the network, with the dropped units being randomly chosen at each iteration. This reduces units co-adaptation, approximates model averaging [34], and can be seen as a form of data augmentation without domain knowledge. For the recurrent layers we adopted the dropout proposed in [35], where the choice of dropped units is kept constant along a sequence.

To speed up the training phase we train our networks with batch normalization layers [36] after every convolutional or fully connected layer. Batch normalization reduces the internal covariate shift—i.e., the distribution of network activations during training—by normalizing a layer output to zero mean and unit variance, using approximate statistics computed on the training mini-batch.

Comparison to other CRNN architectures: The CRNN configuration used in this work has several points of similarity with the network presented in [21] for speech recognition. The main differences are the following:

(i) We do not use any linear projection layer, neither at the end of the CNN part of the CRNN, nor after each recurrent layer. (ii) We use 5x5 kernels in all of our convolutional layers, compared to the 9x9 and 4x3 filters for the first and second layer respectively. (iii) Our architecture has also more convolutional layers (up to 4 instead of 2) and recurrent layers (up to 3 instead of 2). (iv) We use GRU instead of LSTM. (v) We use much longer sequences, up to thousands of steps, compared to 20 steps in [21]. While very long term context is not helpful in speech processing, since words and utterances are quite short in time, in SED there are several events that span over several seconds. (vi) For the experiments on CHiME-Home dataset we incorporate a new max pooling layer (only on time domain) before the output layer. Therefore, if we have N mid-level features for T frames of a context window, we end up with N features for the whole context window to be fed to the output layer.

CNNs and RNNs: It is possible to see CNNs and RNNs as specific instances of the CRNN architecture presented in this section: a CNN is a CRNN with zero recurrent layers, and an RNN is a CRNN with zero convolutional layers. In order to assess the benefits of using CRNNs compared to CNNs or RNNs alone, in Section III we directly compare the three architectures by removing the recurrent or convolutional layer, i.e., CNNs and RNNs respectively.

III. Evaluation

In order to test the proposed method, we run a series of experiments on four different datasets. We evaluate the results by comparing the system outputs to the annotated references. Since we are approaching the task as scene-independent, on each dataset we train a single model regardless of the presence of different acoustic scenes.

A. Datasets and Settings

We evaluate the proposed method on four datasets, one of which is artificially generated as mixtures of isolated sound events, and three are recorded from real-life environments.

While an evaluation performed on real audio data would be ideal, human annotations tend to be somewhat subjective, especially when precise onset and offset are required for overlapping events. For this reason we create our own synthetic dataset—from here onwards referred to as *TUT Sound Events Synthetic 2016* — where we use frame energy based automatic annotation of sound events.

In order to evaluate the proposed method in real-life conditions, we use *TUT Sound Events 2009*. This proprietary dataset contains real-life recordings from 10 different scenes and has been used in many previous works. We also compute and show results on the *TUT Sound Events 2016 development* and *CHiME-Home* dataset, which were used as part of DCASE2016 challenge[1].

a) TUT Sound Events Synthetic 2016 (TUT-SED Synthetic 2016): The primary evaluation dataset consists of synthetic mixtures created by mixing isolated sound events from 16 sound event classes. Polyphonic mixture were created by mixing 994 sound event samples. From the 100 mixtures created, 60% are used for training, 20% for testing and 20% for validation. The total length of the data is 566 minutes. Different instances of the sound events are used to synthesize the training, validation and test partitions. Mixtures were created by randomly selecting event instance and from it, randomly, a segment of length 3-15 seconds. Mixtures do not contain any additional background noise. Dataset creation procedure explanation and metadata can be found in the supporting website for the paper[2].

[1] http://www.cs.tut.fi/sgn/arg/dcase2016/

[2] http://www.cs.tut.fi/sgn/arg/taslp2017-crnn-sed/tut-sed-synthetic-2016

b) TUT Sound Events 2009 (TUT-SED 2009): This dataset, first presented in [37], consists of 8 to 14 binaural recordings from 10 real-life scenes. Each recording is 10 to 30 minutes long, for a total of 1133 minutes. The 10 scenes are: basketball game, beach, inside a bus, inside a car, hallway, office, restaurant, shop, street and stadium with track and field events. A total of 61 classes were defined, including (wind, yelling, car, shoe squeaks, etc.) and one extra class for unknown or rare events. The average number of events active at the same time is 2.53. Event activity annotations were done manually, which introduces a degree of subjectivity. The database has a five-fold cross-validation setup with training, validation and test set split, each consisting of about 60%, 20% and 20% of the data respectively from each scene. The dataset unfortunately can not be made public due to licensing issues, however three $\sim$ 10 minutes samples from the dataset are available at[3].

c) TUT Sound Events 2016 development (TUT-SED 2016): This dataset consists of recordings from two real-life scenes: residential area and home [38]. The recordings are captured each in a different location (i.e., different streets, different homes) leading to a large variability on active sound event classes between recordings. For each location, a 3-5 minute long binaural audio recording is provided, adding up to 78 minutes of audio. The recordings have been manually annotated. In total, there are seven annotated sound event classes for residential area recordings and 11 annotated sound event classes for home recordings. The dataset and metadata is available through[4] and[5].

The four-fold cross-validation setup published along with the dataset [38] is used in the evaluations. Twenty percent of the training set recordings are assigned for validation in the training stage of the neural networks. Since in this work we investigate scene-independent SED, we discard the information about the scene, contrary to the DCASE2016 challenge setup. Therefore, instead of training a separate classifier for each scene, we train a single classifier to be used in all scenes. In TUT-SED 2009 all audio material for a scene was recorded in a single location, whereas TUT-SED 2016 contains multiple locations per scene.

d) CHiME-Home: CHiME-Home dataset [39] consists of 4-second audio chunks from home environments. The annotations are based on seven sound classes, namely child speech, adult male speech, adult female speech, video game / TV, percussive sounds, broadband noise and other identifiable sounds. In this work, we use the same, *refined* setup of CHiME-Home as it is used in audio tagging task in DCASE2016 challenge [40], namely 1946 chunks for development (in four folds) and 846 chunks for evaluation.

The main difference between this dataset and the previous three is that the annotations are made per chunk instead of per frame. Each chunk is annotated with one or multiple labels. In order to adapt our architecture to the lack of frame-level annotations, we simply add a temporal max-pooling layer—that pools the predictions over time—before the output layer for FNN, CNN, RNN and CRNN. CHiME-Home dataset is available at[6].

B. Evaluation Metrics

In this work, segment-based evaluation metrics are used. The segment lengths used in this work are (1): a single time frame (40 ms in this work) and (2): a one-second segment. The segment length for each metric is annotated with the subscript (e.g., $F1_{\mathrm{frm}}$ and $F1_{\mathrm{1sec}}$).

Segment-based F1 score calculated in a single time frame ($F1_{\mathrm{frm}}$) is used as the primary evaluation metric [41]. For each segment in the test set, intermediate statistics, i.e., the number of true positive (*TP*), false positive (*FP*) and false negative (*FN*) entries, are calculated as follows. If an event

1) is detected in one of the frames inside a segment and it is also present in the same segment of the annotated data, that event is regarded as *TP*.
2) is *not* detected in any of the frames inside a segment but it is present in the same segment of the annotated data, that event is regarded as *FN*.
3) is detected in one of the frames inside a segment but it is *not* present in the same segment of the annotated data, that event is regarded as *FP*.

These intermediate statistics are accumulated over the test data and then over the folds. This way, each active instance per evaluated segment has equal influence on the evaluation score. This calculation method is referred to as micro-averaging, and is the recommended method for evaluation of classifier [42]. Precision (P) and recall (R) are calculated from the accumulated intermediate statistics as

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN} \tag{5}$$

These two metrics are finally combined as their harmonic mean, *F1 score*, which can be formulated as

$$F1 = \frac{2 \cdot P \cdot R}{P + R} \tag{6}$$

More detailed and visualized explanation of segment-based F1 score in multi label setting can be found in [41].

The second evaluation metric is segment-based error rate as proposed in [41]. For error rate, intermediate statistics, i.e., the number of substitutions ($\mathbf{s}$), insertions ($\mathbf{i}$), deletions ($\mathbf{d}$) and active classes from annotations ($\mathbf{a}$) are calculated per segment as explained in detail in [41]. Then, the total error rate is calculated as

$$ER = \frac{\sum_{t=1}^{N} \mathbf{s}_t + \sum_{t=1}^{N} \mathbf{i}_t + \sum_{t=1}^{N} \mathbf{d}_t}{\sum_{t=1}^{R} \mathbf{a}_t} \tag{7}$$

where subscript t represents segment index and N is the total number of segments.

Both evaluation metrics are calculated from the accumulated sum for their corresponding intermediate statistics over the segments of the whole test set. If there are multiple scenes in the

[3] http://arg.cs.tut.fi/demo/CASAbrowser/

[4] http://www.cs.tut.fi/sgn/arg/taslp2017-crnn-sed/#tut-sed-2016

[5] https://zenodo.org/record/45759#.WBoUGrPIbRY

[6] https://archive.org/details/chime-home

dataset, evaluation metrics are calculated for each scene separately and then the results are presented as the average across the scenes.

The main metric used in previous works [11], [14], [15] on TUT-SED 2009 dataset differs from the F1 score calculation used in this paper. In previous works, F1 score was computed in each segment, then averaged along segments for each scene, and finally averaged across scene scores, instead of accumulating intermediate statistics. This leads to measurement bias under high class imbalance between the classes and also between folds. However, in order to give a comprehensive comparison of our proposed method with previous works on this dataset, we also report the results with this legacy F1 score in Section IV-B.

For CHiME-Home dataset, equal error rate (EER) has been used as the evaluation metric in order to compare the results with DCASE2016 challenge submissions, where EER has been the main evaluation metric.

C. Baselines

For this work, we compare the proposed method with two recent approaches: the Gaussian mixture model (GMM) of [38] and the feedforward neural network model (FNN) from [15]. GMM has been chosen as a baseline method since it is an established generative modeling method used in many sound recognition tasks [12], [13], [43]. In parallel with the recent surge of deep learning techniques in pattern recognition, FNNs have been shown to vastly outperform GMM based methods in SED [15]. Moreover, this FNN architecture represents a straightforward deep learning method that can be used as a baseline for more complex architectures such as CNN, RNN and the proposed CRNN.

GMM: The first baseline system is based on a binary frame-classification approach, where for each sound event class a binary classifier is set up [38]. Each binary classifier consists of a positive class model and a negative class model. The positive class model is trained using the audio segments annotated as belonging to the modeled event class, and a negative class model is trained using the rest of the audio. The system uses MFCCs as features and a GMM-based classifier. MFCCs are calculated using 40 ms frames with Hamming window and 50% overlap and 40 mel bands. The first 20 static coefficients are kept, and delta and acceleration coefficients are calculated using a window length of 9 frames. The 0th order static coefficient is excluded, resulting in a frame-based feature vector of dimension 59. For each sound event, a positive model and a negative model are trained. The models are trained using expectation-maximization algorithm, using k-means algorithm to initialize the training process and diagonal covariance matrices. The number of parameters for GMM baseline is $3808 * K$, where K is the number of classes. In the detection stage, the decision is based on the likelihood ratio between the positive and negative models for each individual sound class event, with a sliding window of one second. The system is used as a baseline in the DCASE2016 challenge [44], however, in this study the system is used as scene-independent to match the setting of the other methods presented.

FNN: The second baseline system is a deep multi-label FNN with temporal context [15]. As the sound features, 40 log mel band energy features are extracted for each 40 ms time frame with 50% overlap. For the input, consecutive feature vectors are stacked in five vector blocks, resulting in a 100 ms context window. As the hidden layers, two feedforward layers of 1600 hidden units with maxout activation [45] with pool size of 2 units are used. For the output layer, a feedforward layer of K units with sigmoid activation is used to obtain event activity probabilities per context window, where K is the number of classes. The sliding window post-processing of the event activity probabilities in [15] has not been implemented for the baseline experiments in order to make a fair comparison based on classifier architecture for different deep learning methods. The number of parameters in the baseline FNN model is around 1.6 million.

D. Experiments Set-up

Preprocessing: For all neural networks (FNN, CNN, RNN and CRNN) we use log mel band energies as acoustic features. We first compute short-time Fourier transform (STFT) of the recordings in 40 ms frames with 50% overlap, then compute mel band energies through mel filterbank with 40 bands spanning 0 to 22050 Hz, which is the Nyquist rate. After computing the logarithm of the mel band energies, each energy band is normalized by subtracting its mean and dividing by its standard deviation computed over the training set. The normalized log mel band energies are finally split into sequences. During training we use overlapped sequences, i.e., we sample the sub-sequences with a different starting point at every epoch, by moving the starting index by a fixed amount that is not a factor of the sequence length (73 in our experiments). The stride is not equal to 1 in order to have effectively different sub-sequences from one training epoch to the next one. For validation and test data we do not use any overlap.

While finer frequency resolution or different representations could improve the accuracy, our main goal is to compare the architectures. We opted for this setting as it was recently used with very good performance in several works on SED [11], [15].

Neural network configurations: Since the size of the dataset usually affects the optimal network architecture, we do a hyperparameter search by running a series of experiments over predetermined ranges. We select for each network architecture the hyperparameter configuration that leads to the best results on the validation set, and use this architecture to compute the results on the test set.

For TUT-SED Synthetic 2016 and CHiME-Home datasets, we run a hyperparameter grid search on the number of CNN feature maps and RNN hidden units $\{96, 256\}$ (set to the same value); the number of recurrent layers $\{1, 2, 3\}$; and the number of CNN layers $\{1, 2, 3 ,4\}$ with the following frequency max pooling arrangements after each convolutional layer $\{(4), (2, 2), (4, 2), (8, 5), (2, 2, 2), (5, 4, 2), (2, 2, 2, 1), (5, 2, 2, 2)\}$. Here, the numbers denote the number of frequency bands at each max pooling step; *e.g.,* the configuration (5, 4, 2) pools the original

TABLE I
FINAL HYPERPARAMETERS USED FOR THE EVALUATION BASED ON THE VALIDATION RESULTS FROM THE HYPERPARAMETER GRID SEARCH

	TUT-SED Synthetic 2016			TUT-SED 2009			TUT-SED 2016			CHiME-Home		
	CNN	RNN	CRNN	CNN	RNN	CRNN	CNN	RNN	CRNN	CNN	RNN	CRNN
# CNN layers	3	-	3	3	-	3	3	-	3	3	-	4
pool size	(2,2,2)	-	(5,4,2)	(5,4,2)	-	(5,4,2)	(5,4,2)	-	(2,2,2)	(5,4,2)	-	(2,2,2,1)
# RNN layers	-	3	1	-	3	1	-	3	3	-	2	1
# FNN layers	3	2	-	1	4	-	1	4	-	1	1	-
# feature maps/hidden units	256	512	256	256	256	256	256	256	96	256	256	256
sequence length (s)	2.56	5.12	20.48	2.56	20.48	20.48	2.56	20.48	2.56	4	4	4
# Parameters	3.7 M	4.5 M	3.6 M	3.4 M	1.3 M	3.7 M	3.4 M	1.3 M	743 K	3.6 M	690 K	6.1 M

TABLE II
F1 SCORE AND ERROR RATE RESULTS FOR SINGLE FRAME SEGMENTS ($F1_{\mathrm{frm}}$ AND ER_{frm}) AND ONE SECOND SEGMENTS ($F1_{1\mathrm{sec}}$ AND $ER_{1\mathrm{sec}}$)

	TUT-SED Synthetic 2016				TUT-SED 2009				TUT-SED 2016			
Method	$F1_{\mathrm{frm}}$	ER_{frm}	$F1_{1\mathrm{sec}}$	$ER_{1\mathrm{sec}}$	$F1_{\mathrm{frm}}$	ER_{frm}	$F1_{1\mathrm{sec}}$	$ER_{1\mathrm{sec}}$	$F1_{\mathrm{frm}}$	ER_{frm}	$F1_{1\mathrm{sec}}$	$ER_{1\mathrm{sec}}$
GMM [38]	40.5	0.78	45.3	0.72	33.0	1.34	34.1	1.60	14.1	1.12	17.9	1.13
FNN [15]	49.2 ± 0.8	0.68 ± 0.02	50.2 ± 1.4	1.1 ± 0.1	60.9 ± 0.4	0.56 ± 0.01	57.1 ± 0.2	1.1 ± 0.01	26.7 ± 1.4	**0.99 ± 0.03**	**32.5 ± 1.2**	1.32 ± 0.06
CNN	59.8 ± 0.9	0.56 ± 0.01	59.9 ± 1.2	0.78 ± 0.08	64.8 ± 0.2	0.50 ± 0.0	63.2 ± 0.5	0.75 ± 0.02	23.0 ± 2.6	1.02 ± 0.06	26.4 ± 1.9	1.09 ± 0.06
RNN	52.8 ± 1.5	0.6 ± 0.02	57.1 ± 0.9	0.64 ± 0.01	62.4 ± 1.0	0.52 ± 0.01	61.8 ± 0.8	0.55 ± 0.01	**27.6 ± 1.8**	1.04 ± 0.02	29.7 ± 1.4	1.10 ± 0.04
CRNN	**66.4 ± 0.6**	**0.48 ± 0.01**	**68.7 ± 0.7**	**0.47 ± 0.01**	**69.7 ± 0.4**	**0.45 ± 0.0**	**69.3 ± 0.2**	**0.48 ± 0.0**	**27.5 ± 2.6**	**0.98 ± 0.04**	30.3 ± 1.7	**0.95 ± 0.02**

Bold face indicates the best performing method for the given metric.

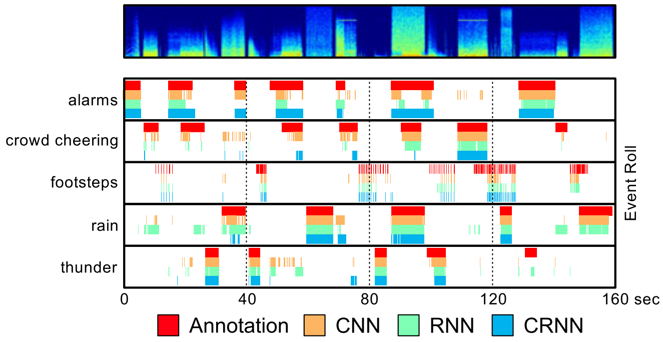

Fig. 3. Annotations and event activity predictions for CNN, RNN and CRNN over a mixture from TUT-SED Synthetic 2016. For clarity, the classes that are not present in the mixture are omitted.

40 bands to one band in three stages: 40 bands → 8 bands → 2 bands → 1 band.

All networks have batch normalization layers after convolutional layers and dropout rate 0.25, which were found to be helpful in preliminary experiments. The output layer consists of a node for each class and has the sigmoid as activation function. In convolutional layers we use filters with shape $(5, 5)$; in recurrent layers we opted for GRU, since preliminary experiments using LSTM yielded similar results and GRU units have a smaller number of parameters. The weights are initialized according to the scheme proposed in [46]. Binary cross-entropy is set as the loss function, and all networks are trained with Adam [47] as gradient descent optimizer, with the default parameters proposed in the original paper.

To evaluate the effect of having both convolutional and recurrent layers in the same architecture, we compare the CRNN with CNNs and RNNs alone. For both CNN and RNN we run the same hyperparameter optimization procedure described for CRNN, replacing recurrent layers with feedforward layers for CNNs, and removing convolutional layers for RNNs while adding feedforward layers before the output layer. This allows for a fair comparison, providing the possibility of having equally deep networks for all three architectures.

After this first optimization process, we use the best CRNNs, CNNs and RNNs to separately test the effect of varying other hyperparameters. More specifically we investigate how performance is affected by variation of the CNN filter shapes and the sequence length. For the CRNN we test filter shapes in the set {(3,3), (5,5), (11,11), (1,5), (5,1), (3,11), (11,3)}, where $(*, *)$ represents the filter lengths in frequency and time axes, respectively. For CRNN and RNN, we test shorter and longer sequences than the initial value of 128 frames, experimenting in the range $\{8, 32, 128, 256, 512, 1024, 2048\}$ frames, which correspond to $\{0.16, 0.64, 2.56, 5.12, 10.24, 20.48, 40.96\}$ seconds respectively. We finally use the hyperparameters that provide the highest validation scores as our final CRNN, CNN and RNN models.

For the other two datasets (TUT-SED 2009 and TUT-SED 2016) we select a group of best performing model configurations on validation data from TUT-SED Synthetic 2016 experiments and to account for the different amount of data we run another smaller hyperparameter search, varying the amount of dropout and the sequence length. Again, we then select the best performing networks on the validation score to compute the test results. The hyperparameters used in the evaluation for all three datasets is presented in Table I.

The event activity probabilities are thresholded at $C = 0.5$, in order to obtain the binary activity matrix used to compute the reference metrics based on the ground truth. All networks are trained until overfitting starts to arise: as a criterion we use early stopping on the validation metric, halting the training if the score is not improving for more than 100 epochs and reverting the weights to the values that best performed on validation.

For feature extraction, the Python library Librosa [48] has been used in this work. For classifier implementations, deep learning package Keras (version 1.1.0) [49] is used with Theano (version 0.8.2) as backend [50]. The networks are trained on NVIDIA Tesla K40t and K80 GPUs.

IV. Results

In this section, we present results for all the datasets and experiments described in Section III. The evaluation of CNN, RNN and CRNN methods are conducted using the hyperparameters given in Table I. All the reported results are computed on the test sets. Unless otherwise stated, we run each neural network based experiment ten times with different random seeds (five times for TUT-SED 2009) to reflect the effect of random weight initialization. We provide the mean and the standard deviation of these experiments in this section. Best performing method is highlighted with bold face in the tables of this section. The methods whose best performance among the ten runs is within one standard deviation of the best performing method is also highlighted with bold face.

The main results with the best performing (based on the validation data) CRNN, CNN, RNN, and the GMM and FNN baselines are reported in Table II. Results are calculated according to the description in Section III-B where each event instance irrespective of the class is taken into account in equal manner. As shown in the table, the CRNNs consistently outperforms CNNs, RNNs and the two baseline methods on all three datasets for the main metric.

A. TUT Sound Events Synthetic 2016

As presented in Table II, CRNN improved by absolute 6.6% and 13.6% on frame-based F1 compared to CNN and RNN respectively for TUT-SED synthetic 2016 dataset. Considering the number of parameters used for each method (see Table I), the performance of CRNN indicates an architectural advantage compared to CNN and RNN methods. All the four deep learning based methods outperform the baseline GMM method. As claimed in [51], this may be due to the capability of deep learning methods to use different subsets of hidden units to model different sound events simultaneously. An example mixture from TUT-SED Synthetic 2016 test set is presented in Fig. 3 with annotations and event activity predictions from CNN, RNN and CRNN.

1) Class-Wise Performance: The class-wise performance with $F1_{\mathrm{frm}}$ metric for CNN, RNN and CRNN methods along with the average and total duration of the classes are presented in Table III. CRNN outperforms both CNN and RNN on almost all classes. It should be kept in mind that each class is likely to appear together with different classes rather than isolated. Therefore the results in Table III present the performance of the methods for each class in a polyphonic setting, as would be the case in a real-life environment. The worst performing class for all three networks is cat meowing, which consists of short, harmonic sounds. We observed that cat meowing samples are mostly confused by baby crying, which has similar acoustic characteristics. Besides, short, non-impulsive sound events are more likely to be masked by another overlapping sound event, which makes their detection more challenging. CRNN performance is considerably better compared to CNN and RNN for gun shot, thunder, bird singing, baby crying and mixer sound events. However, it is hard to make any generalizations on the acoustic characteristics of these events that can explain the superior performance.

TABLE III
$F1_{\mathrm{frm}}$ FOR CNN, RNN AND CRNN FOR EACH CLASS IN TUT-SED SYNTHETIC 2016

Class	avg. (secs)	total (secs)	CNN	RNN	CRNN
glass smash	1.2	621	**57 ± 8.6**	48 ± 2.0	**54 ± 6.7**
gun shot	1.7	534	53 ± 5.9	64 ± 2.3	**73 ± 1.8**
cat meowing	2.1	941	**37 ± 4.6**	29 ± 4.5	**42 ± 3.9**
dog barking	5.0	716	**69 ± 3.3**	51 ± 2.5	**73 ± 3.1**
thunder	5.9	3007	55 ± 3.3	46 ± 2.2	**63 ± 1.9**
bird singing	6.1	2298	44 ± 1.2	41 ± 3.1	**53 ± 2.3**
horse walk	6.4	1614	**46 ± 2.1**	39 ± 2.7	**45 ± 2.4**
baby crying	6.9	2007	46 ± 5.7	46 ± 1.1	**59 ± 3.0**
motorcycle	7.0	3691	**47 ± 3.1**	**44 ± 2.2**	**47 ± 2.7**
footsteps	7.1	1173	41 ± 2.0	34 ± 1.2	**47 ± 1.7**
crowd applause	7.3	3278	**68 ± 1.8**	57 ± 1.5	**71 ± 0.6**
bus	7.8	3464	60 ± 2.0	55 ± 2.5	**66 ± 2.4**
mixer	7.9	4020	62 ± 5.6	57 ± 6.4	**82 ± 2.7**
crowd cheering	8.1	4825	72 ± 2.9	64 ± 2.7	**77 ± 1.1**
alarms	8.2	4405	**64 ± 2.2**	50 ± 5.3	**66 ± 2.9**
rain	8.2	3975	**71 ± 2.0**	59 ± 2.6	**72 ± 1.9**

TABLE IV
$F1_{\mathrm{frm}}$ FOR ACCURACY VS. CONVOLUTION FILTER SHAPE FOR TUT-SED SYNTHETIC 2016 DATASET

Filter shape	(3,3)	(5,5)	(11,11)	(1,5)	(5,1)	(3,11)	(11,3)
$F1_{\mathrm{frm}}$	67.2	**68.3**	62.6	28.5	60.6	67.4	61.2

$(*, *)$ represents filter lengths in frequency and time axis, respectively.

2) Effects of Filter Shape: The effect of the convolutional filter shape is presented in Table IV. Since these experiments were part of the hyperparameter grid search, each experiment is conducted only once. Small kernels, such as (5,5) and (3,3), were found to perform the best in the experiments run on this dataset. This is consistent with the results presented in [31] on a similar task. The very low performance given for the filter shape (1,5) highlights the importance of including multiple frequency bands in the convolution when spectrogram based features are used as input for the CRNN.

3) Number of Parameters vs. Accuracy: The effect of number of parameters on the accuracy is investigated in Fig. 4. The points in the figure represent the test accuracy with $F1_{\mathrm{frm}}$ metric for the hyperparameter grid search experiments. Each experiment is conducted one time only. Two observations can be made from the figure. For the same number of parameters, CRNN has a clear performance advantage over CNN and RNN. This

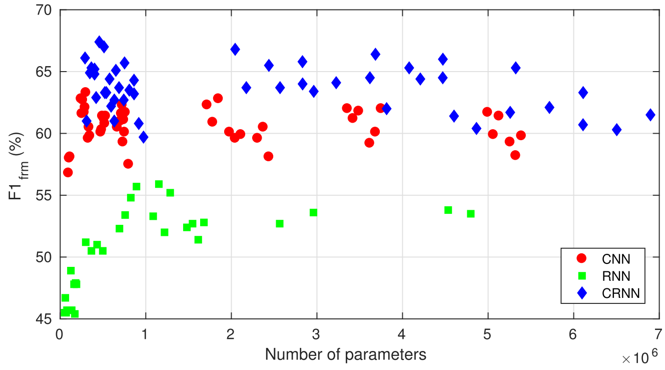

Fig. 4. Number of parameters vs. accuracy for CNN, RNN and CRNN.

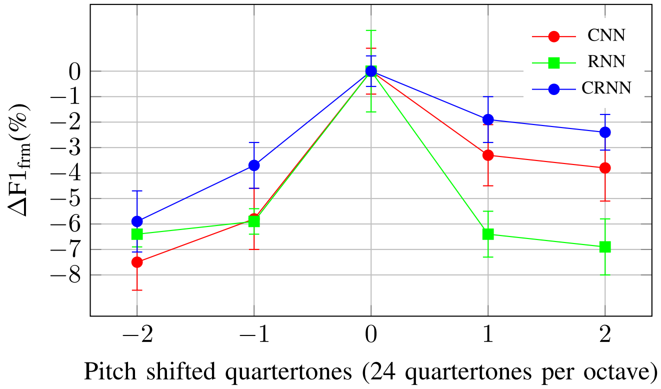

Fig. 5. Absolute accuracy change vs. pitch-shifting over ± 2 quartertones for CNN, RNN and CRNN.

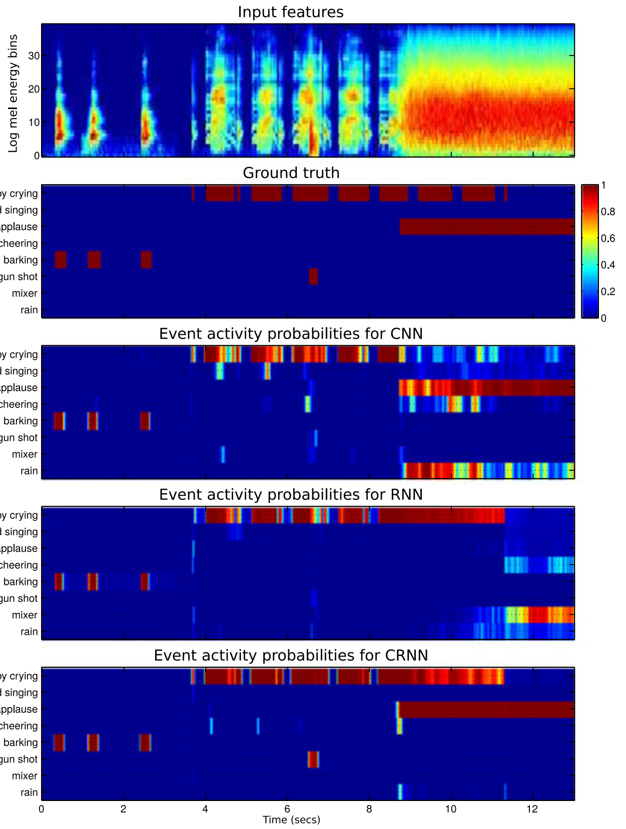

Fig. 6. Input features, ground truth and event activity probabilities for CNN, RNN and CRNN from a sequence of test examples from TUT-SED synthetic 2016.

indicates that the high performance of CRNN can be explained with the architectural advantage rather than the model size. In addition, there can be a significant performance shift for the same type of networks with the same number of parameters, which means that a careful grid search on hyperparameters (e.g., shallow with more hidden units per layer vs. deep with less hidden units per layer) is crucial in finding the optimal network structure.

4) Frequency Shift Invariance: Sound events may exhibit small variations in their frequency content. In order to investigate the robustness of the networks to small frequency variations, pitch shift experiments are conducted and the absolute changes in frame-based F1 score are presented in Fig. 5. For these experiments, each network is first trained with the original training data. Then, using Librosa's pitch-shift function, the pitch for the mixtures in the test set is shifted by ± 2 quartertones. The test results show a significant absolute drop in accuracy for RNNs when the frequency content is shifted slightly. As expected, CNN and CRNN are more robust to small changes in frequency content due to the convolution and max-pooling operations. However, accuracy decrease difference between the methods diminishes for negative pitch shift, for which the reasons should be further investigated. It should be also noted that RNN has the lowest base accuracy, so it is relatively more affected for the same amount of absolute accuracy decrease (see Table II).

5) Closer Look on Network Outputs: A comparative study on the neural network outputs, which are regarded as event activity probabilities, for a 13-second sequence of the test set is presented in Fig. 6. For the parts of the sequence where *dog barking* and *baby crying* appear alone, all three networks successfully detect these events. However, when a *gun shot* appears overlapping with *baby crying*, only CRNN can detect the *gun shot* although there is a significant change in the input feature content. This indicates the efficient modeling of the *gun shot* by CRNN which improves the detection accuracy even in polyphonic conditions. Moreover, when *crowd applause* begins to appear in the signal, it almost completely masks *baby crying*, as it is evident from the input features. CNN correctly detects *crowd applause*, but misses the masked *baby crying* in this case, and RNN ignores the significant change in features and keeps detecting *baby crying*. RNN's insensitivity to the input feature change can be explained with its input gate not passing through new inputs to recurrent layers. On the other hand, CRNN correctly detects both events and almost perfectly matches the ground truth along the whole sequence.

B. TUT-SED 2009

For a comprehensive comparison, results with different methods applied to the same cross-validation setup and published over the years are shown in Table V. The main metric used in

TABLE V
RESULTS FOR TUT-SED 2009 BASED ON THE LEGACY F1

Method	Legacy $F1_{1\text{sec}}$
HMM multiple Viterbi decoding* [24]	20.4
NMF-HMM* [25]	36.7
NMF-HMM + stream elimination* [25]	44.9
GMM* [38]	34.6
Coupled NMF* [14]	57.8
FNN [15]	63.0
BLSTM [11]	64.6
CNN	63.9 ± 0.4
RNN	62.2 ± 0.8
CRNN	**69.1 ± 0.4**

Methods marked with ⋆ are trained in scene-dependent setting.

these previous works is averaged over folds, and may be influenced by distribution of events in the folds (see Section III-B). In order to allow a direct comparison, we have computed all metrics in the table the same way.

First published systems were scene-dependent, where information about the scene is provided to the system and separate event models are trained for each scene [14], [24], [25]. More recent work [11], [15], as well as the current study, consist of scene-independent systems. Methods [24], [25] are HMM based, using either multiple Viterbi decoding stages or NMF pre-processing to do polyphonic SED. In contrast, the use of NMF in [14] does not build explicit class models, but performs coupled NMF of spectral representation and event activity annotations to build dictionaries. This method performs polyphonic SED through direct estimation of event activities using learned dictionaries.

The results on the dataset show significant improvement with the introduction of deep learning methods. CRNN has significantly higher performance than previous methods [14], [24], [25], [38], and it still shows considerable improvement over other neural network approaches.

C. TUT-SED 2016

The CRNN and RNN architectures obtain the best results in terms of framewise *F1*. The CRNN outperforms all the other architectures for *ER* framewise and on 1-second blocks. While the FNN obtains better results on the 1-second block *F1*, this happens at the expense of a very large 1-second block *ER*.

For all the analyzed architectures, the overall results on this dataset are quite low compared to the other datasets. This is most likely due the fact that TUT-SED 2016 is very small and the sounds events occur sparsely (i.e., a large portion of the data is silent). In fact, when we look at class-wise results (unfortunately not available due to space restrictions), we noticed a significant performance difference between the classes that are represented the most in the dataset (e.g. bird singing and car passing by, $F1_{\text{frm}}$ around 50%) and the least represented classes (e.g., cupboard and object snapping, $F1_{\text{frm}}$ close to 0%). Some other techniques might be applied to improve the accuracy of systems trained on such small datasets, e.g. training a network on a larger dataset and then retraining the output layer on the smaller dataset (transfer learning), or incorporating unlabeled data to the learning process (semi-supervised learning).

TABLE VI
EQUAL ERROR RATE (EER) RESULTS FOR CHIME-HOME DEVELOPMENT AND EVALUATION DATASETS

Method	Development EER	Evaluation EER
Lidy *et al.* [52]	17.8	16.6
Cakir *et al.* [53]	17.1	16.8
Yun *et al.* [54]	17.6	17.4
CNN	**12.6 ± 0.5**	**10.7 ± 0.6**
RNN	16.0 ± 0.3	13.8 ± 0.4
CRNN	**13.0 ± 0.3**	**11.3 ± 0.6**
CNN (no batch norm)	15.1 ± 1.7	11.9 ± 1.0

D. CHiME-Home

The results obtained on CHiME-Home are reported in Table VI. For all of our three architectures there is a significant improvement over the previous results reported on the same dataset on the DCASE2016 challenge, setting new state-of-the-art results.

After the first series of experiments the CNN obtained slightly better results compared to the CRNN. The CRNN and CNN architecture used are almost identical, with the only exception of the last recurrent (GRU) layer in the CRNN being replaced by a fully connected layer followed by batch normalization. In order to test if the improvement in the results was due to the absence of recurrent connections or to the presence of batch normalization, we run again the same CNN experiments removing the normalization layer. As shown in the last row of VI, over 10 different random initializations the average EER increased to values above those obtained by the CRNN.

E. Visualization of Convolutional Layers

Here we take a peek at the representation learned by the networks. More specifically, we use the technique described in [55] to visualize what kind of patterns in the input data different neurons in the convolutional layers are looking for. We feed the network a random input whose entries are independently drawn from a Gaussian distribution with zero mean and unit variance. We choose one neuron in a convolutional layer, compute the gradient of its activation with respect to the input, and iteratively update the input through gradient ascent in order to increase the activation of the neuron. If the gradient ascent optimization does not get stuck into a weak local maximum, after several updates the resulting input will strongly activate the neuron. We run the experiment for several convolutional neurons in the CRNN networks trained on TUT-SED Synthetic 2016 and TUT-SED 2009, halting the optimization after 100 updates. In Fig. 7 we present a few of these inputs for several neurons at different depth. The figure confirms that the convolutional filters have specialized into finding specific patterns in the input. In addition, the complexity of the patterns looked for by the filters seems to increase as the layers become deeper.

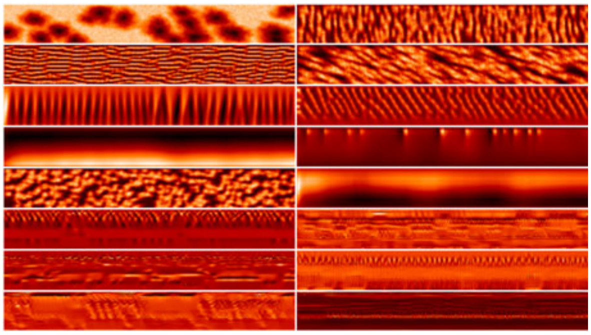

Fig. 7. Two columns of crops from input patterns that would strongly activate certain neurons from different layers of the CRNN. On the horizontal axis is time, on the vertical axis mel bands. On both columns the rows 1 and 2 are from neurons in the first convolutional layer, rows 3 to 5 from the second, and rows from 6 to 8 from the third.

V. Conclusion

In this work, we proposed to apply a CRNN—a combination of CNN and RNN, two complementary classification methods—on a polyphonic SED task. The proposed method first extracts higher level features through multiple convolutional layers (with small filters spanning both time and frequency) and pooling in frequency domain; these features are then fed to recurrent layers, whose features in turn are used to obtain event activity probabilities through a feedforward fully connected layer. In CRNN, CNN's capability to learn local translation invariant filters and RNN's capability to model short and long term temporal dependencies are gathered in a single classifier. The evaluation results over four datasets show a clear performance improvement for the proposed CRNN method compared to CNN, RNN, and other established methods in polyphonic SED.

Despite the improvement in performance, we identify a limitation to this method. As presented in TUT-SED 2016 results in Table II, the performance of the proposed CRNN (and of the other deep learning based methods) strongly depends on the amount of available annotated data. TUT-SED 2016 dataset consists of 78 minutes of audio of which only about 49 minutes are annotated with at least one of the classes. When the performance of CRNN for TUT-SED 2016 is compared to the performance on TUT-SED 2009 (1133 minutes) and TUT-SED Synthetic 2016 (566 minutes), there is a clear performance drop both in the absolute performance and in the relative improvement with respect to other methods. Dependency on large amounts of data is a common limitation of current deep learning methods.

The results we observed in this work, and in many other classification tasks in various domains, prove that deep learning is definitely worth further investigation on polyphonic SED. As a future work, semi-supervised training methods can be investigated to overcome the limitation imposed by small datasets. Transfer learning [56], [57] could be potentially applied with success in this setting: by first training a CRNN on a large dataset (such as TUT-SED Synthetic 2016), the last feedforward layer can then be replaced with random weights and the network fine-tuned on the smaller dataset.

Another issue worth investigating would be a detailed study over the activations from different stages of the proposed CRNN method. For instance, a class-wise study over the higher level features extracted from the convolutional layers might give an insight on the common features of different sound events. Finally, recurrent layer activations may be informative on the degree of relevance of the temporal context information for various sound events.

Acknowledgment

The authors wish to acknowledge CSC IT Center for Science, Finland, for computational resources. The paper has a supporting website at http://www.cs.tut.fi/sgn/arg/taslp2017-crnn-sed/.

References

[1] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento, "Reliable detection of audio events in highly noisy environments," *Pattern Recognit. Lett.*, vol. 65, pp. 22–28, 2015.

[2] S. Goetze, J. Schroder, S. Gerlach, D. Hollosi, J.-E. Appell, and F. Wallhoff, "Acoustic monitoring and localization for social care," *J. Comput. Sci. Eng.*, vol. 6, no. 1, pp. 40–50, 2012.

[3] J. Salamon and J. P. Bello, "Feature learning with deep scattering for urban sound analysis," in *Proc. 2015 23rd Eur. Signal Process. Conf.*. 2015, pp. 724–728.

[4] Y. Wang, L. Neves, and F. Metze, "Audio-based multimedia event detection using deep recurrent neural networks," in *Proc. 2016 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 2742–2746.

[5] D. Stowell and D. Clayton, "Acoustic event detection for multiple overlapping similar sources," in *Proc. 2015 IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2015, pp. 1–5.

[6] J. W. Dennis, "Sound event recognition in unstructured environments using spectrogram image processing," Ph.D. dissertation, School Comput. Eng., Nanyang Technol. Uni., Singapore, 2014.

[7] O. Gencoglu, T. Virtanen, and H. Huttunen, "Recognition of acoustic events using deep neural networks," in *Proc. Eur. Signal Process. Conf.*, 2014, pp. 506–510.

[8] H. Zhang, I. McLoughlin, and Y. Song, "Robust sound event recognition using convolutional neural networks," in *Proc. 2015 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 559–563.

[9] H. Phan, L. Hertel, M. Maass, and A. Mertins, "Robust audio event recognition with 1-max pooling convolutional neural networks," in *Proc. Interspeech*, 2016, pp. 3653–3657.

[10] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *Proc. Int. Workshop Mach. Learn. Signal Process.*, 2015, pp. 1–6.

[11] G. Parascandolo, H. Huttunen, and T. Virtanen, "Recurrent neural networks for polyphonic sound event detection in real life recordings," in *Proc. 2016 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 6440–6444.

[12] L.-H. Cai, L. Lu, A. Hanjalic, H.-J. Zhang, and L.-H. Cai, "A flexible framework for key audio effects detection and auditory context inference," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 14, no. 3, pp. 1026–1039, May 2006.

[13] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *Proc. Eur. Signal Process. Conf.*, 2010, pp. 1267–1271.

[14] A. Mesaros, O. Dikmen, T. Heittola, and T. Virtanen, "Sound event detection in real life recordings using coupled matrix factorization of spectral representations and class activity annotations," in *Proc. 2015 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 151–155.

[15] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Polyphonic sound event detection using multilabel deep neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, 2015, pp. 1–7.

[16] E. Cakir, E. Ozan, and T. Virtanen, "Filterbank learning for deep neural network based polyphonic sound event detection," in *Proc. Int. Joint Conf. Neural Netw.*, 2016, pp. 3399–3406.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Adv. Neural Inform. Process. Syst.*, 2012, pp. 1097–1105.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[20] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. 2013 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2013, pp. 6645–6649.

[21] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Proc. 2015 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2015, pp. 4580–4584.

[22] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1724–1734.

[23] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3128–3137.

[24] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Context-dependent sound event detection," *EURASIP J. Audio, Speech, Music Process.*, vol. 2013, no. 1, pp. 1–13, 2013.

[25] T. Heittola, A. Mesaros, T. Virtanen, and M. Gabbouj, "Supervised model training for overlapping sound events based on unsupervised source separation," in *Proc. Int. Conf. Acoust, Speech, Signal Process.*, 2013, pp. 8677–8681.

[26] O. Dikmen and A. Mesaros, "Sound event detection using non-negative dictionaries learned from annotated overlapping events," in *Proc. 2013 IEEE Workshop Appl. Signal Process. Audio Acoust*, 2013, pp. 1–4.

[27] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[28] D. Amodei *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *Proc. 33rd Int. Conf. Mach. Learn.*, 2016, pp. 173–182.

[29] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform CLDNNs," in *Proc. Interspeech*, 2015, pp. 1–5.

[30] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *Proc. Int. Conf. Acoust, Speech, Signal Process.*, 2017 (submitted).

[31] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 24, no. 5, pp. 927–939, May 2016.

[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[33] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *Proc. 8th Workshop Syntax, Semantics Structure Statistical Translation*, 2014, pp. 103–112.

[34] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[35] Y. Gal, "A theoretically grounded application of dropout in recurrent neural networks," in *Adv. Neural Inform. Process. Syst.*, 2016, pp. 1019–1027.

[36] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[37] T. Heittola, A. Mesaros, A. Eronen, and T. Virtanen, "Audio context recognition using audio event histograms," in *Proc. 18th Eur. Signal Process. Conf.*, 2010, pp. 1272–1276.

[38] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," in *Proc. 24th Eur. Signal Process. Conf.*, 2016, pp. 1128–1132.

[39] P. Foster, S. Sigtia, S. Krstulovic, J. Barker, and M. D. Plumbley, "Chime-home: A dataset for sound source recognition in a domestic environment," in *Proc. IEEE Workshop Appl. Signal Process. Audio Acoust.*, 2015, pp. 1–5.

[40] T. Heittola. DCASE2016 challenge—Audio tagging. Accessed on Sep. 2016. [Online]. Available: http://www.cs.tut.fi/sgn/arg/dcase2016/task-audio-tagging

[41] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Appl. Sci.*, vol. 6, no. 6, 2016, Art. no. 162.

[42] G. Forman and M. Scholz, "Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement," *ACM SIGKDD Explorations Newslett.*, vol. 12, no. 1, pp. 49–57, 2010.

[43] W.-H. Cheng, W.-T. Chu, and J.-L. Wu, "Semantic context detection based on hierarchical audio models," in *Proc. 5th ACM SIGMM Int. Workshop Multimedia Inf. Retrieval*, 2003, pp. 109–115.

[44] T. Heittola, A. Mesaros, and T. Virtanen, "DCASE2016 baseline system," 2016. [Online]. Available: https://github.com/TUT-ARG/DCASE2016-baseline-system-python

[45] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio, "Maxout networks." in *Proc. Int. Conf. Mach. Learn.*, 2013, vol. 28, pp. 1319–1327.

[46] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.

[47] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations Workshop*, 2015.

[48] B. McFee *et al.*, "librosa: Audio and music signal analysis in Python," in *Proc. 14th Python Sci. Conf.*, 2015.

[49] F. Chollet, "Keras," 2016. [Online]. Available: https://github.com/fchollet/keras

[50] T. T. D. Team *et al.*, "Theano: A python framework for fast computation of mathematical expressions," arXiv:1605.02688, 2016.

[51] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[52] T. Lidy and A. Schindler, "CQT-based convolutional neural networks for audio scene classification and domestic audio tagging," Tech. Rep., DCASE2016 Workshop, Budapest, Hungary, Sep. 2016.

[53] E. Cakir, T. Heittola, and T. Virtanen, "Domestic audio tagging with convolutional neural networks," Tech. Rep., DCASE2016 Workshop, Budapest, Hungary, Sep. 2016.

[54] S. Yun, S. Kim, S. Moon, J. Cho, and T. Kim, "Discriminative training of GMM parameters for audio scene classification," Tech. Rep., DCASE2016 Workshop, Budapest, Hungary, Sep. 2016.

[55] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *Proc. Int. Conf. Learn. Representations Workshop*, 2014.

[56] Y. Bengio *et al.*, "Deep learning of representations for unsupervised and transfer learning," *ICML Unsupervised Transfer Learn.*, vol. 27, pp. 17–36, 2012.

[57] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Adv. Neural Inform. Process. Syst.*, 2014, pp. 3320–3328.

Emre Çakır received the B.Sc. degree in electrical and electronics engineering from Middle East Technical University, Ankara, Turkey, in 2013 and the M.Sc. degree in information technology from Tampere University of Technology (TUT), Tampere, Finland, in 2015. Since February 2014, he has been with the Audio Research Group in TUT, where he currently working toward the Ph.D. degree. His main research interests include sound event detection in real-life environments and deep learning.

Giambattista Parascandolo received the B.Sc. degree from the Department of Mathematics, University of Rome Tor Vergata, Roma, Italy, in 2013, and the M.Sc. degree in information technology from Tampere University of Technology (TUT), Tampere, Finland, in 2015. He is a Project Researcher in the Audio Research Group in TUT, where he has been since February, 2015. His main research interests include deep learning and machine learning.

Toni Heittola received the M.Sc. degree in information technology from Tampere University of Technology (TUT), Tampere, Finland, in 2004. He is currently working toward the Ph.D. degree at TUT. His main research interests include sound event detection in real-life environments, sound scene classification, and audio content analysis.

Heikki Huttunen received the Ph.D. degree in signal processing at Tampere University of Technology (TUT), Tampere, Finland, in 1999. He is currently a university Lecturer in the Department of Signal Processing, TUT. He is an author of more than 100 research articles on signal and image processing and analysis. His research interests include optical character recognition, deep learning, and pattern recognition and statistics.

Tuomas Virtanen received the M.Sc. and Doctor of Science degrees in information technology from Tampere University of Technology (TUT), Tampere, Finland, in 2001 and 2006, respectively. He is known for his pioneering work on single-channel sound source separation using nonnegative matrix factorization based techniques, and their application to noise-robust speech recognition, music content analysis, and audio event detection. In addition to the above topics, his research interests include content analysis of audio signals in general and machine learning. He has received the IEEE Signal Processing Society 2012 best paper award. He is an Academy Research Fellow and an adjunct professor in the Department of Signal Processing, TUT.