Sound Event Detection with Depthwise Separable and Dilated Convolutions

Konstantinos Drossos
Audio Research Group
Tampere University
Tampere, Finland
konstantinos.drossos@tuni.fi

Stylianos I. Mimilakis
Semantic Music Technologies Group
Fraunhofer-IDMT
Ilmenau, Germany
mis@idmt.fraunhofer.de

Shayan Gharib
Audio Research Group
Tampere University
Tampere, Finland
shayan.gharib@tuni.fi

Yanxiong Li
School of Electronic & Information Engineering
South China University of Technology
Guangzhou, China
eeyxli@scut.edu.cn

Tuomas Virtanen
Audio Research Group
Tampere University
Tampere, Finland
tuomas.virtanen@tuni.fi

***Abstract*—State-of-the-art sound event detection (SED) methods usually employ a series of convolutional neural networks (CNNs) to extract useful features from the input audio signal, and then recurrent neural networks (RNNs) to model longer temporal context in the extracted features. The number of the channels of the CNNs and size of the weight matrices of the RNNs have a direct effect on the total amount of parameters of the SED method, which is to a couple of millions. Additionally, the usually long sequences that are used as an input to an SED method along with the employment of an RNN, introduce implications like increased training time, difficulty at gradient flow, and impeding the parallelization of the SED method. To tackle all these problems, we propose the replacement of the CNNs with depthwise separable convolutions and the replacement of the RNNs with dilated convolutions. We compare the proposed method to a baseline convolutional neural network on a SED task, and achieve a reduction of the amount of parameters by 85% and average training time per epoch by 78%, and an increase the average frame-wise F$_1$ score and reduction of the average error rate by 4.6% and 3.8%, respectively.**

***Index Terms*—sound event detection, depthwise separable convolution, dilated convolution**

I. Introduction

Sound event detection (SED) is the task of identifying onsets and offsets of target class activities in general audio signals [1]. A typical SED scenario involves a method which takes as an input an audio signal, and outputs temporal activity for target classes like “car passing by”, “footsteps”, “people talking”, “gunshot”, etc [1], [2]. The time resolution of the activity of classes can vary among different methods and datasets, but typically is used 0.02 sec [1]–[4]. Also, activities of classes can overlap (polyphonic SED) or not (monophonic SED). SED can be employed in a wide range of applications, like wildlife monitoring and bird activity detection [5], [6], home monitoring [7], [8], autonomous vehicles [9], [10], and surveillance [11], [12].

Current deep learning-based SED methods can be viewed as a composition of three functions. The first function is a feature extractor, usually implemented by convolutional neural network (CNN) blocks (i.e. a CNN followed by a non-linearity, and normalization and sub-sampling processes), which provides frequency shift invariant features of the input audio signal [1]. The second function, implemented by a recurrent neural network (RNN), models long temporal context and inter- and intra-class patterns in the output of the feature extractor (i.e. the first function) [2]. Finally, the third function, which is an affine transform followed by a sigmoid non-linearity (in the case of polyphonic detection), performs the classification. In [1] is described a widely adopted method that conforms to the above mentioned scheme, consisting of three CNN blocks followed by an RNN and a classifier. This method is termed as convolutional recurrent neural networks (CRNN) and has been used in a variety of audio processing tasks, like music emotion recognition [13], sound event detection and localization [14], bird activity detection [5], [6], and SED [1].

The typical amount of parameters of the CRNN is around 3.5 M, and the sequence length of the input audio and the output predictions is 1024 frames. Because an RNN is used, the CRNN method cannot be parallelized (i.e. split between different processing units, e.g. GPUs). The 1024 time-frame length of the output sequence can be considered long enough to create computational problems at the calculation of the gradient, due to the RNN (e.g. gated recurrent units, GRU, or long short-term memory, LSTM). Reduction of the number of parameters of an SED model would allow the method to be fit for systems with restricted resources (e.g. embedded systems) and the training time would decrease (resulting in

Part of the computations leading to these results was performed on a TITAN-X GPU donated by NVIDIA. Part of the research leading to these results has received funding from the European Research Council under the European Union’s H2020 Framework Programme through ERC Grant Agreement 637422 EVERYSOUND. Stylianos I. Mimilakis is supported in part by the German Research Foundation (AB 675/2-1, MU 2686/11-1). The authors wish to acknowledge CSC-IT Center for Science, Finland, for computational resources.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

faster experimentation and optimization). Also, removing the RNN would allow the method to be split between different processing units, would have more efficient training, and the amount of parameters could be further reduced.

In this paper, we propose the replacement of the CNNs and the RNN. In particular, we propose the employment of depthwise separable convolutions [15]–[18] instead of typical CNNs, resulting in a considerable decrease of the parameters for the learned feature extractor. We also propose the replacement of the RNN with dilated convolutions [19]–[21]. This allows modeling long temporal context, but reduces the amount of parameters, eliminates the gradient problems due to the usually long employed sequences (e.g. 1024-frame long), and allows for parallelization of the model [22], [23].

Similar approaches have been proposed in [24], [25] and in the code of the YAMNET system, available online[1]. Specifically, in [24] is proposed a method using a series of dilated convolutions as a feature extractor, instead of typical CNNs. The output of the last dilated convolution is given as an input to an RNN, which does not lift any of the shortcomings of using RNNs in SED. In [25] is proposed a system for sound event tagging and based on the MobileNet [26], using one 1D typical CNN layer, followed by 13 layers of depthwise separable convolutions. The output of the last depthwise separable convolution layer is sub-sampled and used as an input to a classifier for sound event tagging. YAMNET is also based on the MobileNet [26], using depthwise separable convolutions. The amount of parameters of the YAMNET amounts to 3.7M. In both [25] and YAMNET there was not a specific module for taking into account the modeling of the longer temporal context in the input audio (e.g. like an RNN or a dilated convolution).

To evaluate the impact of our proposed changes, we employ a typical method for SED that is based on stacked CNNs and RNNs [1], and a freely available SED dataset, the TUTSED Synthetic 2016 [27]. Our results show that with our proposed changes we reduce the amount of parameters by 85% and the average time per epoch need for training by 78% (measured on the same GPU), while we increase the frame-wise F_1 score by 4.6% and decrease the error rate by 3.8%. The rest of the paper is as follows. In Section II we briefly present the baseline approach and in Section III is our proposed method. Section IV explains the evaluation set-up of our method and the obtained results are presented in Section V. Section VI concludes the paper and proposes future research directions.

II. Baseline approach

The baseline approach accepts as an input a sequence of T audio feature vectors $\mathbf{X} \in \mathbb{R}^{T\times N}$, with each vector having N features, and associated target output corresponding to the activities of C classes $\mathbf{Y} \in \{0,1\}^{T\times C}$. $\mathbf{X}$ is given as an input to a learnable feature extractor f_{cnn}, consisting of cascaded 2D CNN blocks. Each block has a 2D CNN followed by a non-linearity, a normalization process, and a feature sub-sampling process. The output of f_{cnn} is given as an input to a temporal pattern identification module f_{rnn}, which consists of a GRU RNN. f_{rnn} is followed by a classifier f_{cls}, which is an affine transform followed by a sigmoid non-linearity. The output of f_{cls} for each of the T feature vectors is the predicted activities for each of the C classes $\hat{\mathbf{Y}} \in [0,1]^{T\times C}$. During inference process, the activities $\hat{\mathbf{Y}}$ are further binarized using a threshold of 0.5.

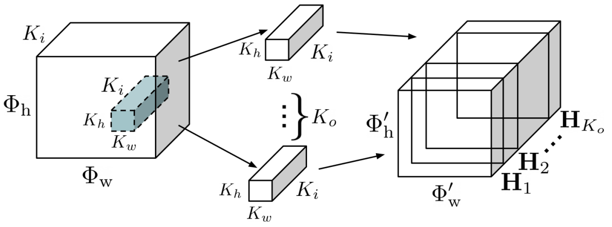


Fig. 1: A typical process for a CNN. Each of the K_o kernels, of size $K_\text{i} \times K_\text{h} \times K_\text{w}$, is convolved with K_i input matrices of $T \times N$ size. The output is K_o different matrices of $T' \times N'$ size. Bias is omitted for clarity.

A. Learnable feature extractor based on CNNs

The learnable feature extractor of the baseline approach consists of three CNN blocks, each block having a typical 2D CNN followed by a rectified linear unit (ReLU), a batch normalization process, and max-pooling operation across the dimension of features. A typical 2D CNN consist of K_o kernels $\mathbf{K} \in \mathbb{R}^{K_\text{i}\times K_\text{h}\times K_\text{w}}$ and bias vectors $\mathbf{b} \in \mathbb{R}^{K_\text{i}}$, where K_i and K_o are the number of input and output channels of the CNN, and K_h and K_w are the height and width of the kernel of each channel. Each kernel $\mathbf{K}$ is applied to the input $\mathbf{\Phi} \in \mathbb{R}^{K_\text{i}\times \Phi_\text{h}\times \Phi_\text{w}}$ of the 2D CNN to obtain the output $\mathbf{H} \in \mathbb{R}^{K_o\times \Phi'_\text{h}\times \Phi'_\text{w}}$ of the 2D CNN, as

$$\begin{aligned}\mathbf{H}_{k_\text{o},\phi'_h-K_\text{h},\phi'_w-K_\text{w}} =&(\mathbf{K}_{k_\text{o}} * \mathbf{\Phi})(k_\text{i}, \phi_\text{h}-k_\text{h}, \phi_\text{w}-k_\text{w})\\ =&\sum_{k_\text{i}}^{K_\text{i}}\sum_{k_\text{h}}^{K_\text{h}}\sum_{k_\text{w}}^{K_\text{w}} \mathbf{\Phi}_{k_\text{i},\phi_\text{h}-k_\text{h},\phi_\text{w}-k_\text{w}}\mathbf{K}_{k_\text{o},k_\text{h},k_\text{w}},\end{aligned} \tag{1}$$

where $*$ is the convolution operator with *unit stride* and zero padding. The above application of $\mathbf{K}$ onto $\mathbf{\Phi}$ leads to learning and extracting spatial and cross-channel information from the input features $\mathbf{\Phi}$ [16], and has a computational complexity of $O(K_\text{o}\cdot K_\text{i}\cdot K_\text{h}\cdot K_\text{w}\cdot \Phi_\text{h}\cdot \Phi_\text{w})$ [16]–[18]. Additionally, the amount of learnable parameters of the 2D CNN (omitting bias) is $K_\text{i}\cdot K_\text{o}\cdot K_\text{h}\cdot K_\text{w}$. Figure 1 illustrates the above operation.

In each CNN block of the feature extractor, the output of the 2D CNN is followed by ReLU, batch normalization, and max-pooling operations. The output of the max-pooling operation is given as an input to the next CNN block. The output of the third CNN block $\mathbf{H}^3 \in \mathbb{R}^{K_o^3\times \Phi_h^3\times \Phi_w^3}$, with K_o^3 to be the output channels of the third CNN (denoted with the superscript 3), is reshaped to $\mathbf{H}^{cnn} \in \mathbb{R}^{\Phi_h^{cnn}\times \Phi_w^{cnn}}$, where $\Phi_h^{cnn} = \Phi_h^3$ and $\Phi_w^{cnn} = K_o^3 \cdot \Phi_w^3$. $\mathbf{H}^{cnn}$ is given as an input to the GRU of the f_{rnn}.

[1] https://github.com/tensorflow/models/tree/master/research/audioset/yamnet

B. Gated recurrent unit for long temporal context identification

The output features $\mathbf{H}^{cnn}$ of f_{cnn} are likely to include multi-scale contextual information, encoding long temporal patterns and inter- and intra-class activity [2]. To exploit this information, the baseline approach utilizes f_{rnn}, which is a GRU that gets as an input the $\mathbf{H}^{cnn}$. The input and output dimensionality of f_{rnn} the same and equal to Φ^{cnn}_{w}.

In particular, the GRU of f_{rnn} takes as an input the output of the last CNN block of the baseline approach $\mathbf{H}^{cnn}$ and processes each row ϕ^{cnn}_{h} according to the equations mentioned in the original paper [28]. The output of f_{rnn}, $\mathbf{H}^{rnn} \in [-1, 1]^{\Phi^{cnn}_{h} \times \Phi^{cnn}_{w}}$ is given as an input to the classifier f_{cls}.

C. Classifier, loss, and optimization

The classifier f_{cls} gets as an input the output of f_{rnn}, $\mathbf{H}^{rnn}$. f_{cls} consists of a learnable affine transform with shared weights through time, followed by a sigmoid non-linearity. The output of f_{cls} is the output of the CRNN method, which is

$$\hat{\mathbf{Y}} = f_{\text{cls}}(\mathbf{H}^{rnn}). \tag{2}$$

f_{cnn}, f_{rnn}, and f_{cls} are jointly optimized by minimizing the cross-entropy loss between $\hat{\mathbf{Y}}$ and $\mathbf{Y}$.

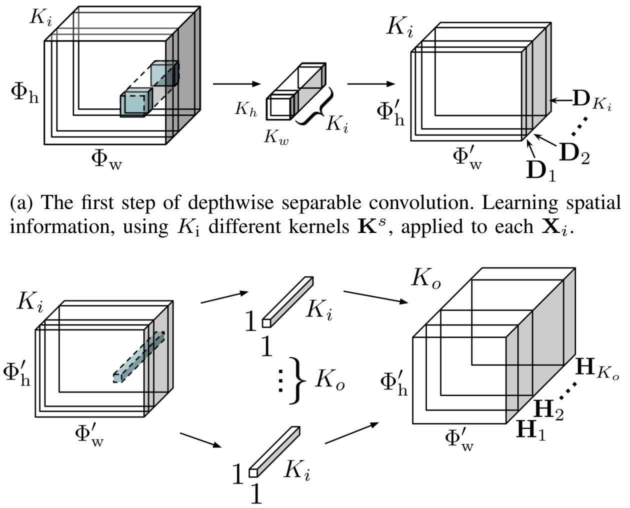
III. Proposed approach

In our method we replace the f_{cnn} and f_{rnn} with different types of convolutions. We replace the f_{cnn} with depthwise separable convolutions, which result in smaller amount of parameters and increased performance [18], [26], [29]–[31]. Additionally, we replace the f_{rnn} with dilated convolutions, which have smaller amount of parameters, are based on CNNs, and can model long temporal context [19]–[21].

Specifically, our method also accepts as an input $\mathbf{X} \in \mathbb{R}^{T\times N}$ and the associated annotations for the activities of classes $\mathbf{Y} \in \{0, 1\}^{T\times C}$. $\mathbf{X}$ is given as an input to a learnable feature extractor f_{dws}, consisting of cascaded 2D depthwise separable CNN (DWS-CNN) blocks. Each block has a 2D CNN based on depthwise separable convolution followed by a non-linearity, a normalization process, and a feature subsampling process. The output of f_{dws} is given as an input to a temporal pattern identification module f_{dil}, which consists of a 2D CNN based on dilated convolution (DIL-CNN). f_{dil} is followed by a classifier f_{cls}, which is the same classifier as in the baseline approach. The output of f_{cls} for each of the T feature vectors is the predicted activities for each of the C classes $\hat{\mathbf{Y}} \in [0, 1]^{T\times C}$. Similarly to the baseline, during the inference process, the activities $\hat{\mathbf{Y}}$ are further binarized using a threshold of 0.5.

A. Learnable feature extractor based on depthwise separable convolutions

Based on [15] and for our f_{dws}, we employ the factorization of the spatial and cross-channel learning process described by Eq (1). We replace the 2D CNNs of the CRNN method with 2D DWS-CNNs, closely following the DWS-CNNs presented for the MobileNets model [26] and the hyper-parameters used in the CRNN architecture [1]. Instead of using $\mathbf{\Phi}$ in a convolution with a single kernel $\mathbf{K}$ in order to learn spatial and cross-channel information, we apply, in series, two convolutions (i.e. the output of the first is the input to the second) using two different kernels. This factorization technique is termed as depthwise separable convolution, has been adopted to a variety of architectures for image processing (like the XCeption, GoogleLeNet, Inception, and MobileNets models), and has been proven to increase the performance while reducing the amount of parameters [18], [26], [29]–[31].



(a) The first step of depthwise separable convolution. Learning spatial information, using K_{i} different kernels $\mathbf{K}^s$, applied to each $\mathbf{X}_i$.

(b) The second step of depthwise separable convolution. Learning cross-channel information using K_{o} different kernels $\mathbf{K}^z$.

Fig. 2: The process of depthwise separable convolution. Bias is omitted for clarity.

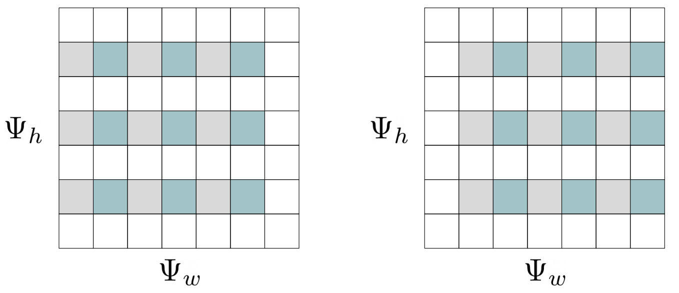
Firstly, we apply K_{i} kernels $\mathbf{K}^{\text{s}} \in \mathbb{R}^{K_{\text{h}}\times K_{\text{w}}}$ to each $\mathbf{\Phi}_{k_{\text{i}}}$ in order learn the spatial relationships of features in $\mathbf{X}$ as

$$\begin{aligned}\mathbf{D}_{k_{\text{i}},t-K_{\text{h}},n-K_{\text{w}}} =&(\mathbf{K}^{\text{s}}_{k_{\text{i}}} * \mathbf{X}_{k_{\text{i}}})(t-K_{\text{h}}, n-K_{\text{w}})\\ =&\sum_{k_{\text{h}}}^{K_{\text{h}}}\sum_{k_{\text{w}}}^{K_{\text{w}}} \mathbf{X}_{k_{\text{i}},t-k_{\text{h}},n-k_{\text{w}}}\mathbf{K}^{\text{s}}_{k_{\text{i}},k_{\text{h}},k_{\text{w}}},\end{aligned} \tag{3}$$

where $\mathbf{D}_{k_{\text{i}}} \in \mathbb{R}^{\Phi'{}_h\times\Phi'_w}$. Then, we utilize K_{o} kernels $\mathbf{k}^{\text{z}}_{k_o} \in \mathbb{R}^{K_{\text{i}}}$, with $\mathbf{K} = \{\mathbf{k}^{\text{z}}_1, \mathbf{k}^{\text{z}}_2, \ldots, \mathbf{k}^{\text{z}}_{K_o}\}$, and we apply them $\mathbf{D} = \{\mathbf{D}_1, \ldots, \mathbf{D}_{K_{\text{i}}}\}$, in order learn the cross-channel relationships, as

$$\mathbf{H}_{k_{\text{o}},\phi'_h,\phi'_w} = \sum_{k_{\text{i}}}^{K_{\text{i}}} \mathbf{D}_{k_{\text{i}},\phi'_h,\phi'_w}\mathbf{K}^{\text{z}}_{k_{\text{o}},k_i}. \tag{4}$$

The resulting computational complexity and amount of parameters (omitting bias), for both processes in Eq. (3) and (4), are $O(K_{\text{h}} \cdot K_{\text{w}} \cdot K_{\text{i}} \cdot \Phi_h \cdot \Phi_w + K_{\text{i}} \cdot K_{\text{o}} \cdot \Phi'_h \cdot \Phi'_w)$ and $K_i \cdot K_h \cdot K_w + K_i \cdot K_o$, respectively. Thus, the computational complexity [26] and amount of parameters are both reduced by $K_{\text{o}}^{-1} + (K_{\text{h}} \cdot K_{\text{w}})^{-1}$ times. The process of depthwise convolution is illustrated in Figure 2, with the first step in Figure 2a and the second in Figure 2b.



(a) Calculation of $\mathbf{H}'_{k'_o,\psi'_h,\psi'_w}$ (b) Calculation of $\mathbf{H}'_{k_o,\psi'_h,\psi'_w+1}$

Fig. 3: Illustration of the process described in Eq. (5) using $\xi_h = \xi_w = 2$ and calculating two consecutive elements of $\mathbf{H}'_{k'_o,\psi'_h}$. Squares coloured with cyan signify the elements participating at the calculations for $\mathbf{H}'_{k'_o,\psi'_h,\psi'_w}$, and coloured with grey are the elements for $\mathbf{H}'_{k'_o,\psi'_h,\psi'_w-1}$.

According to the baseline approach, we use three blocks of DWS-CNNs, where each block consists of a DWS-CNN, followed by a rectified linear unit (ReLU), a batch normalization process, and a max pooling operation across the dimension of features Φ_w. $\mathbf{H}^3$ is the output of the third DWS-CNN block, which is given as an input to f_{dil}.

B. Dilated convolutions

Contrary to the baseline approach, we employ f_{dil} in order to exploit the long temporal patterns in $\mathbf{H}^3$. f_{dil} is based on 2D dilated convolutions, which are capable to aggregate and learn multi-scale information and have been used previously in image processing tasks [19]–[21].

A dilated 2D CNN (DIL-CNN) consists of K'_o kernels $\mathbf{K}' \in \mathbb{R}^{K'_\text{i} \times K'_\text{h} \times K'_\text{w}}$ and bias vectors $\mathbf{b}' \in \mathbb{R}^{K'_o}$. Similarly to the typical CNN described in Section III-A, K'_i and K'_o are the input and output channels of the DIL-CNN, and K'_h and K'_w are the height and width of the kernel for each channel. Each $\mathbf{K}'$ is applied to the input of DIL-CNN $\mathbf{\Psi} \in \mathbb{R}^{K'_i \times \Psi_h \times \Psi_w}$ to obtain the output $\mathbf{H}' \in \mathbb{R}^{K'_o \times \Psi'_h \times \Psi'_w}$ of the DIL-CNN as

$$\begin{aligned}\mathbf{H}'_{k'_\text{o},\psi'_h-k'_\text{h},\psi'_w-k'_\text{w}} &= (\mathbf{K}'_{k'_\text{o}} * \mathbf{\Psi})(k'_\text{i}, \psi_h - \xi_\text{h} \cdot k'_\text{h}, \psi_w - \xi_\text{w} \cdot k'_\text{w}) \\ &= \sum_{k'_\text{i}}^{K'_\text{i}} \sum_{k'_\text{h}}^{K'_\text{h}} \sum_{k'_\text{w}}^{K'_\text{w}} \mathbf{\Psi}_{k_{\text{i}'},\psi_\text{h}-\xi_\text{h}\cdot k_\text{h},\psi_\text{w}-\xi_\text{w}\cdot k'_\text{w}} \mathbf{K}'_{k'_\text{o},k'_\text{h},k'_\text{w}},\end{aligned} \tag{5}$$

where $\xi_h, \xi_w \in \mathbb{N}^\star$ are the dilation rates for the K'_h and K'_w dimensions of $\mathbf{K}'$. It should be denoted that for $\xi_h = \xi_w = 1$, Eq. (5) boils down to Eq. (1), i.e. a typical convolution with no dilation.

The dilation rates, ξ_h and ξ_w, multiply the index that is used for accessing elements from $\mathbf{\Psi}$. This allows a scaled aggregation of contextual information at the output of the operation [21]. Practically, this means that the resulting features computed by using DIL-CNN (i.e. $\mathbf{H}'$) are calculated from a bigger area, resulting into modelling longer temporal context. The growth of the area that $\mathbf{H}'$ is calculated from, is equal $\xi_h \cdot \xi_w$. The process described by Eq. (5) is illustrated in Figure 3.

We use DIL-CNN to replace the recurrent neural networks that efficiently model long temporal context and inter- and intra-class activities for SED. Specifically, our f_{dil} has $K'_i = K_o$, takes as an input the output of f_{dws}, $\mathbf{H}^L$, and outputs $\mathbf{H}'$, as

$$\mathbf{H}' = f_{dil}(\mathbf{H}^L), \text{ and} \tag{6}$$

$$\mathbf{H}^{dil} = BNorm(ReLU(\mathbf{H}')). \tag{7}$$

Finally, $\mathbf{H}^{dil}$ is reshaped to $\Psi'_h \times (K_o \cdot \Psi'_w)$ and given as an input to the classifier of our method, which is the f_{cls} of the baseline approach.

IV. Evaluation setup

To assess the performance of each of the proposed replacements and their combination, we employ a freely available SED dataset and we compare the performance of the CRNN and each of our proposed replacements. The code for all the models and the evaluation process described in this paper, is freely available online[2].

A. Baseline system and models

We employ four different models, Model$_\text{base}$, Model$_\text{dw}$, Model$_\text{dil}$, and Model$_\text{dnd}$. Model$_\text{base}$ is our main baseline and consists of three CNN blocks, followed by a GRU, and a linear layer acting as a classifier. Each CNN block consists of a CNN with 256 channels, square kernel shape of $\{5, 5\}$, stride of $\{1, 1\}$, and padding of $\{2, 2\}$, followed by a ReLU, a batch normalization, a max pooling, and a dropout of 0.25 probability. The max pooling operations have kernels and stride of $\{1, 5\}$, $\{1, 4\}$, and $\{1, 2\}$. The GRU has 256 input and output features, and the classifier has 256 input and 16 output features.

For our second model, Model$_\text{dws}$, we replace the CNN blocks at CRNN with f_{dws}, so we can assess the benefit of using DWS-CNNs instead of typical 2D CNNs. To minimize the factors that will have an impact to possible differences between our proposed method and the employed baseline, for our f_{dws} we adopted the same kernel shapes, strides, and padding for the $\mathbf{K}_\text{s}$ kernels, as in the Model$_\text{base}$. That is, all K_o, K_h, and K_w of f_{dws} have the same values as the corresponding ones in Model$_\text{base}$. The same stands true for stride and padding, and all hyper-parameters of max-pooling operations.

At the third model, Model$_\text{dil}$, we replace the GRU in Model$_\text{base}$ with the f_{dil}, so we can assess the benefit of using DIL-CNN instead of an RNN. Since there are no previous studies using DIL-CNNs as a replacement for RNNs and for SED, we opt to keep the same amount of channels at the DWS-CNNs and perform a grid search on K'_h, K'_w, and ξ_h. Specifically, we employ four different kernel shapes $(K'_h, K'_w) \in \{(3, 3), (5, 5), (7, 7)\}$. We denote the different shapes of kernels with an exponent, e.g. Model^3_dil for the model

[2] https://github.com/dr-costas/dnd-sed

having an f_{dil} with a kernel of shape of $\{3, 3\}$, or $\text{Model}_{\text{dnd}}^{7}$ for the model having f_{dws} and an f_{dil} of kernel with shape $\{7, 7\}$. Because we want to assess the effect of using a different time-resolution for capturing inter- and intra-event patterns with the DIL-CNN, we use $\xi_w = 1$ and $\xi_h \in \{1, 10, 50, 100\}$. That is, we apply dilation only on the time dimension and not on the dimension of features. Though, to keep the time dimension intact (i.e. to have $\Psi'_h = T$), we use zero padding at the time dimension. Specifically, we use a padding equal to ξ_h for kernel shape of $(3, 3)$, a padding equal to $2 \cdot \xi_h$ for $(5, 5)$ kernel, $3 \cdot \xi_h$ for the $(7, 7)$ kernel, and $5 \cdot \xi_h$ for the $(11, 11)$ kernel. We use no padding at the feature dimension for the f_{dil}. Must be noted that when $\xi_h = 1$ then f_{dil} is a typical 2D CNN and, thus, we also assess the effect of replacing the RNN with a typical 2D CNN. We also denote the employed dilation in the exponent, e.g. $\text{Model}_{\text{dil}}^{3|50}$ or $\text{Model}_{\text{dnd}}^{7|1}$.

Finally, the $\text{Model}_{\text{dnd}}$ is our complete proposed method, where we replace both the typical CNN blocks and the GRU from the $\text{Model}_{\text{base}}$, with the f_{dws} and f_{dil}, respectively. For complete assessment of our proposed method, we follow the same grid search on on K'_h, K'_w, and ξ_h, as we perform for $\text{Model}_{\text{dil}}$.

B. Dataset and metrics

We use the TUT-SED Synthetic 2016 dataset, which is freely available online[3] and has been employed in multiple previous work on SED [1], [2], [32]. TUT-SED Synthetic consists of 100 mixtures of around eight minutes length with isolated sound events from 16 classes, namely alarms & sirens, baby crying, bird singing, bus, cat meowing, crowd applause, crowd cheering, dog barking, footsteps, glass smash, gun shot, horse walk, mixer, motorcycle, rain, and thunder. The mixtures are split to training, validation, and testing split by 60%, 20%, and 20%, respectively. The maximum polyphony of the dataset is 5. From each mixture we extract multiple sequences of $T = 1024$ vectors, having $N = 40$ log-mel band energies and using a hamming window of ≈ 0.02 sec, with 50% overlap. As the evaluation metrics we use F_1 score and error rate (ER), similarly to the original paper of CRNN and previous work on SED [1], [2], [32]. Both of the metrics are calculate on a per-frame basis (i.e. for every $t = 1, 2, \ldots, T$). Additionally, we keep a record of the training time per epoch for each model and for all repetitions of the optimization process, by measuring the elapsed time between the start and the end of each epoch.

C. Training and testing procedures

We optimize the parameters of all models (under all sets of hyper-parameters) using the training split of the employed dataset, the Adam optimizer with values for hyper-parameters (i.e. β_1, β_2, and ϵ) as proposed in the original paper [33], a batch size of 16, and cross-entropy loss. After one full iteration over the training split (i.e. one epoch), we employ the same loss and measure its value on the validation split. We stop the optimization process if the loss on the validation split does not improve for 30 consecutive epochs and we keep the values of the parameters of the model from the epoch yielding the lowest validation loss. Finally, we assess the performance of each model using the testing split and the employed metrics (i.e. F_1 and ER).

In order to have an objective assessment of the impact of our proposed method, we repeat 10 times the optimization for every model, following the above described process. Then, we calculate the average and standard deviation of the above mentioned metrics, i.e., F_1 score and error rate (ER). In addition to this, we report the number of parameters (N_P) and the necessary mean training time per epoch ($\overline{E}_T$), i.e., a full iteration throughout the whole training split. All presented experiments performed on an NVIDIA Pascal V100 GPU.

V. Results and discussion

In Table I are the results from all conducted experiments, organized in two groups. The first one is termed as SED performance and regards the performance of each model and set of hyper-parameters for the SED task (i.e. F_1 and ER). The second group, termed as computational performance, considers the number of parameters and average time necessary for training (N_P and $\overline{E}_T$), for each model and each set of hyper-parameters. The STD of F_1 and ER is in the range of 0 to 0.02 and omitted for clarity.

The baseline CRNN system, i.e. $\text{Model}_{\text{base}}$, seems to perform better in classification only from $\text{Model}_{\text{dnd}}^{7|100}$. In every other case, $\text{Model}_{\text{base}}$ yields worse classification performance. This indicates that our proposed changes can, in general, result to better classification performance when compared to the baseline system. Regarding the computational performance, can be seen that there are specific sets of hyper-parameters that result to models with more parameters from $\text{Model}_{\text{base}}$. Specifically, $\text{Model}_{\text{dil}}^{3}$ and $\text{Model}_{\text{dil}}^{5}$ with all ξ_h, have more parameters than $\text{Model}_{\text{base}}$. This increase in N_P, though, is not attributed on the difference of the amount of parameters between f_{dil} of $\text{Model}_{\text{dil}}$ and the GRU of $\text{Model}_{\text{base}}$, but on the amount of parameters that the classifier has. In the case of $\text{Model}_{\text{base}}$, the output of the GRU had dimensions of 1024×256. The classifier has shared weights through time, thus the amount of its input features is 256. But, in the case of $\text{Model}_{\text{dil}}^{3}$ and $\text{Model}_{\text{dil}}^{5}$, the dimensionality of the input to the classifier, i.e. $\mathbf{H}^{dil}$, is $256 \times 1024 \times \Psi'_w$, where Ψ'_w is inverse proportional to the size of the kernel of f_{dil}. After reshaping $\mathbf{H}^{dil}$ to $1024 \times (256 \cdot \Psi'_w)$, the amount of input features to the classifier is $K_o \cdot \Psi'_w$, which is considerably bigger than the $\text{Model}_{\text{base}}$ case, i.e. 1024×256. Finally, $\text{Model}_{\text{base}}$ needs more time (on average) per epoch compared to any other model and set of hyper-parameters in Table I. This clearly indicates that all of the proposed changes have a positive impact on the needed time per epoch, even in the case where N_P is bigger.

Comparing the impact of each of the changes (i.e. $\text{Model}_{\text{dws}}$ versus $\text{Model}_{\text{dil}}$), we can see that adopting DWS-CNN can significantly increase the SED performance, yielding better F_1 and ER compared to $\text{Model}_{\text{base}}$ and $\text{Model}_{\text{dil}}$ (except $\text{Model}_{\text{dil}}^{5|10}$). Additionally, $\text{Model}_{\text{dws}}$ yields the lowest ER in

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

TABLE I: Quantitative results from evaluating the effect of using depth-wise separable (Model$_{\text{dwd}}$) or dilated (Model$_{\text{dil}}$), or both (Model$_{\text{dnd}}$) convolutions as modifications to the baseline CRNN architecture (Model$_{\text{base}}$). Average (mean) values of the F1 score ($\overline{F}_1$, higher the better) and the error rate ($\overline{\text{ER}}$, lower the better) are reported over the ten repetitions. The number of parameters is denoted by N_P and the average (and standard deviation, STD) time, in seconds, required for an epoch by $\overline{E}_T$ (±STD). N/A denotes a non applicable parameterization. Bold faced elements denote the best reported performance for classification and computational performance.

				SED Performance		**Computational Performance (mean±STD)**	
Model$_*$	DWS	ξ_{h}	$(K'_{\text{h}}, K'_{\text{w}},)$	$\overline{F}_1$	$\overline{\text{ER}}$	N_P	$\overline{E}_T$
base	×	N/A	N/A	0.5z	0.54	3.68M	49.4 (±11.8)
dil	×	1	(3 × 3)	0.60	0.54	3.81M	14.1 (±0.06)
	×	10	(3 × 3)	0.61	0.53	3.81M	14.1 (±0.11)
	×	50	(3 × 3)	0.62	0.51	3.81M	14.1 (±0.07)
	×	100	3 × 3)	0.61	0.53	3.81M	14.5 (±0.08)
	×	1	(5 × 5)	0.60	0.54	3.81M	20.7 (±0.09)
	×	10	(5 × 5)	0.63	0.51	3.81M	18.2 (±0.25)
	×	50	(5 × 5)	0.60	0.52	3.81M	18.5 (±0.07)
	×	100	(5 × 5)	0.58	0.56	3.81M	18.5 (±0.08)
	×	1	(7 × 7)	0.60	0.54	3.64M	12.2 (±0.06)
	×	10	(7 × 7)	0.62	0.52	3.64M	12.2 (±0.07)
	×	50	(7 × 7)	0.61	0.52	3.64M	12.4 (±0.07)
	×	100	(7 × 7)	0.58	0.57	3.64M	12.4 (±0.07)
dws	✓	N/A	(3 × 3)	0.62	0.50	0.62M	46.9 (±4.81)
dnd	✓	1	(3 × 3)	0.59	0.54	0.74M	13.0 (±0.06)
	✓	10	(3 × 3)	0.62	0.51	0.74M	13.0 (±0.06)
	✓	50	(3 × 3)	0.61	0.53	0.74M	13.0 (±0.10)
	✓	100	(3 × 3)	0.60	0.53	0.74M	13.4 (±0.08)
	✓	1	(5 × 5)	0.59	0.55	0.74M	20.1 (±3.63)
	✓	10	(5 × 5)	0.62	0.52	0.74M	17.0 (±0.24)
	✓	50	(5 × 5)	0.62	0.52	0.74M	17.4 (±0.01)
	✓	100	(5 × 5)	0.58	0.56	0.74M	17.4 (±0.01)
	✓	1	(7 × 7)	0.60	0.54	0.58M	11.4 (±4.45)
	✓	10	(7 × 7)	**0.63**	**0.50**	**0.58M**	**11.1** (±0.06)
	✓	50	(7 × 7)	0.61	0.53	0.58M	11.2 (±0.17)
	✓	100	(7 × 7)	0.58	0.57	0.58M	11.3 (±0.11)

total, but not the highest F$_1$. Furthermore, Model$_{\text{dws}}$ has $N_P =$ 0.62 M, significantly less than any Model$_{\text{dil}}$ and the Model$_{\text{base}}$. The decrease in the amount of parameters and the increase in the performance when using the f_{dws} is in accordance with previous studies that adopted DWS-CNN [18], [26], [29]–[31]. Focusing on the Model$_{\text{dil}}$, can be observed that the usage of dilation increases the classification performance. Specifically, in all kernel shapes, the $\xi_h = 1$ (i.e. no dilation) yields the lowest F$_1$ and highest ER. Also, it is apparent that for $\xi_h \geq 50$ the classification performance decreases.

Finally, when both f_{dws} and f_{dil} are combined (i.e. Model$_{\text{dnd}}$) it seems that there is a drop in the performance (compared to Model$_{\text{dws}}$) for the (3, 3) and (5, 5) kernel shapes and for all ξ_h. But, for the case of Model$^{7|10}_{\text{dnd}}$, there is the highest F$_1$ score and by 0.02 second ER. Additionally, the specific Model$^{7|10}_{\text{dnd}}$ model needs the less average time per epoch and belongs to the group of models with the less parameters.

VI. Conclusion and future work

In this paper, we proposed the adoption of depthwise separable and dilated convolutions based 2D CNNs, as a replacement of usual 2D CNNs and RNN layers in typical SED methods. To evaluate our proposed changes, we conducted a series of experiments, assessing each replacement in separate and also their combination. We used a widely adopted method and a freely available SED dataset. Our results showed that when both DWS-CNN and DIL-CNN are used, instead of usual CNNs and RNNs, respectively, the resulting method has considerably better classification performance, the amount of parameters decreases by 80%, and the average needed time (for training) per epoch decreases by 72

Although we conducted a grid search of the hyper-parameters, the proposed method is likely not fine tuned for the task of SED. Further study is needed in order to fine tune the hyper-parameters and yield the maximum classification performance for the task of SED.

References

[1] E. Çakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 25, no. 6, pp. 1291–1303, Jun. 2017.

[2] K. Drossos, S. Gharib, P. Magron, and T. Virtanen, "Language modelling for sound event detection with teacher forcing and scheduled sampling," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, Oct. 2019.

[3] S. Kapka and M. Lewandowski, "Sound source detection, localization and classification using consecutive ensemble of CRNN models," in *Workshop of Detection and Classification of Acoustic Scenes and Events (DCASE)*, Oct. 2019.

[4] F. Grondin, J. Glass, I. Sobieraj, and M. D. Plumbley, "Sound event localization and detection using CRNN on pairs of microphones," in *Workshop of Detection and Classification of Acoustic Scenes and Events (DCASE)*, Oct. 2019.

[5] E. Çakir, S. Adavanne, G. Parascandolo, K. Drossos, and T. Virtanen, "Convolutional recurrent neural networks for bird audio detection," in *2017 25th European Signal Processing Conference (EUSIPCO)*, Aug. 2017, pp. 1744–1748.

[6] S. Adavanne, K. Drossos, E. Çakir, and T. Virtanen, "Stacked convolutional and recurrent neural networks for bird audio detection," in *2017 25th European Signal Processing Conference (EUSIPCO)*. IEEE, 2017, pp. 1729–1733.

[7] N. Turpault, R. Serizel, A. Shah, and J. Salamon, "Sound event detection in domestic environments with weakly labeled data and soundscape synthesis," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, Oct. 2019.

[8] R. M. Alsina-Pagès, J. Navarro, F. Alías, and M. Hervás, "homeSound: Real-time audio event detection based on high performance computing for behaviour and surveillance remote monitoring," *Sensors (Basel)*, vol. 17, no. 4, 2017.

[9] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Surrey-CVSSP system for DCASE2017 challenge task4," DCASE2017 Challenge, Tech. Rep., Sep. 2017.

[10] D. Lee, S. Lee, Y. Han, and K. Lee, "Ensemble of convolutional neural networks for weakly-supervised sound event detection using multiple scale input," DCASE2017 Challenge, Tech. Rep., Sep. 2017.

[11] L. Marchegiani and P. Newman, "Listening for sirens: Locating and classifying acoustic alarms in city scenes," *ArXiv*, vol. abs/1810.04989, 2018.

[12] Y. Fu, K. Xu, H. Mi, H. Wang, D. Wang, and B. Zhu, "A mobile application for sound event detection," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*, Oct. 2019.

[13] M. Malik, S. Adavanne, K. Drossos, T. Virtanen, D. Ticha, and R. Jarina, "Stacked convolutional and recurrent neural networks for music emotion recognition," in *14th Sound & Music Computing Conference (SMC-17)*, Jul. 2017, pp. 208–213.

[14] S. Adavanne, A. Politis, J. Nikunen, and T. Virtanen, "Sound event localization and detection of overlapping sources using convolutional recurrent neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 1, p. 34–48, Mar 2019. [Online]. Available: http://dx.doi.org/10.1109/JSTSP.2018.2885636

[15] L. Sifre, "Rigid-motion scattering for image classification," Ph.D. dissertation, Ecole Polytechnique, CMAP, 2014.

[16] J. Guo, Y. Li, W. Lin, Y. Chen, and J. Li, "Network decoupling: From regular to depthwise separable convolutions," in *British Machine Vision Conference (BMVC)*, Sep. 2018.

[17] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *ArXiv*, vol. abs/1704.04861, 2017.

[18] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 1800–1807.

[19] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A real-time algorithm for signal analysis with the help of the wavelet transform," in *Wavelets*, J.-M. Combes, A. Grossmann, and P. Tchamitchian, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 286–297.

[20] M. J. Shensa, "The discrete wavelet transform: wedding the a trous and mallat algorithms," *IEEE Transactions on Signal Processing*, vol. 40, no. 10, pp. 2464–2482, Oct 1992.

[21] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *International Conference on Learning Representations (ICLR)*, May 2016.

[22] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation," in *Computer Vision – ECCV 2016 Workshops*, G. Hua and H. Jégou, Eds. Cham: Springer International Publishing, 2016, pp. 47–54.

[23] Y. He and J. Zhao, "Temporal convolutional networks for anomaly detection in time series," *Journal of Physics: Conference Series*, vol. 1213, p. 042050, jun 2019. [Online]. Available: https://doi.org/10.1088%2F1742-6596%2F1213%2F4\%2F042050

[24] Y. Li, M. Liu, K. Drossos, and T. Virtanen, "Sound event detection via dilated convolutional recurrent neural networks," in *in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2020.

[25] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, and X. Serra, "Audio tagging with noisy labels and minimal supervision," in *Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, Oct. 2019.

[26] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *ArXiv*, vol. abs/1704.04861, 2017.

[27] "TUT-SED Synthetic 2016," http://www.cs.tut.fi/sgn/arg/taslp2017-crnn-sed/tut-sed-synthetic-2016, accessed: 2019-12-10.

[28] K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.

[29] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 1–9.

[30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 2818–2826.

[31] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, p. 448–456.

[32] G. Huang, T. Heittola, and T. Virtanen, "Using sequential information in polyphonic sound event detection," in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, September 2018.

[33] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference for Learning Representations*, May 2015.