CONDITIONED TIME-DILATED CONVOLUTIONS FOR SOUND EVENT DETECTION

Konstantinos Drossos, Stylianos I. Mimilakis, Tuomas Virtanen

1 Audio Research Group, Tampere University, Tampere, Finland
{firstname.lastname}@tuni.fi
2 Fraunhofer-IDMT, Ilmenau, Germany
mis@idmt.fraunhofer.de

ABSTRACT

Sound event detection (SED) is the task of identifying sound events along with their onset and offset times. A recent, convolutional neural networks based SED method, proposed the usage of depthwise separable (DWS) and time-dilated convolutions. DWS and time-dilated convolutions yielded state-of-the-art results for SED, with considerable small amount of parameters. In this work we propose the expansion of the time-dilated convolutions, by conditioning them with jointly learned embeddings of the SED predictions by the SED classifier. We present a novel algorithm for the conditioning of the time-dilated convolutions which functions similarly to language modelling, and enhances the performance of the these convolutions. We employ the freely available TUT-SED Synthetic dataset, and we assess the performance of our method using the average F₁ score and average per-frame error rate, over the 10 experiments. We achieve an increase of 2% (from 0.63 to 0.65) at the average F₁ score (the higher the better) and a decrease of 3% (from 0.50 to 0.47) at the error rate (the lower the better).

Index Terms— sound event detection, depthwise separable convolutions, dilated convolutions, conditioned dilated convolutions

1. INTRODUCTION

Sound event detection (SED) is a typical machine listening task, and is the identification of activities of different sound events (e.g. “people speaking”, “dog barking”) together with the detection of their onset and offset times. This is usually implemented by identifying the active sound events in a sequence of short-time windows of the recording (e.g. 0.02 seconds), effectively detecting when these sound events are starting and ending [1,2]. Typical SED methods are usually based on deep neural networks (DNNs), and consist of a learnable feature extractor and a module for detecting temporal patterns on top of the output of the feature extractor [2,4]. Typically in the literature, the feature extractor is implemented using convolutional neural networks (CNNs), and the identification of the temporal patterns using recurrent neural networks (RNNs). Such set-up is seen in the wide-spread convolutional-recurrent neural networks (CRNN) architecture [5] and its adoption to different SED and audio signal processing tasks and methods, e.g. [6,7,8,9,10].

Recently, there is an interest in re-examining the usage of the CNNs and RNNs in SED and audio classification, in general. There are published papers focusing on the tasks of sound event tagging and SED, where the CNNs are been replaced by depthwise separable (DWS) convolutions [2,9]. DWS convolutions are a factorized version of the typical convolutions in the CNNs, by first learning the spatial information and then processing the cross-channel patterns [12,13]. This factorization of the convolution carried out by typical CNNs, leads to considerable fewer trainable parameters and smaller memory footprint of the CNNs, reducing the computational complexity by a factor of $K_w^{-1} + (K_h K_w)^{-1}$, where $K_h$ and $K_w$ are the height and width of the CNN kernel, and $K_w$ are the output channels of the CNN [2]. This reduction of the computational complexity, renders the SED and sound event tagging systems more fit for deployment in devices with restricted resources, e.g. embedded and wearable devices.

The employment of the DWS convolutions has not only yielded DNNs with smaller memory footprint, but in some cases also boosted the performance of the DNN [2,9]. For example, YAMNET [14] is a sound event tagging system, using specifically DWS convolutions and focusing on having a small amount of total parameters. In [11] is presented another system for sound event tagging, implemented with 13 layers of DWS convolutions. Though, nor the system in [11], neither YAMNET, have implemented any explicit functionality for detecting long temporal patterns. Such long temporal patters and apparent for different classes or between different classes. For example, “car passing by” is more likely to appear before or after a “car horn”, and a “water tap running” is an event with longer duration that water “flushing” or “splashing”.

Dilated convolutions have been proposed as way to increase the capabilities of learning long temporal patterns for CNNs [15]. In a nutshell, the kernel of a dilated convolution is expanded, having distance between its elements. This allows the kernel of the dilated convolution to be applied on every other $N$ elements (where $N$ is the dilation factor) of the input patch to the kernel, increasing the receptive field of the kernel without increasing its parameters [15]. Dilation of the kernel can be applied to any combination of its dimensions (e.g. dilation only at the dimension of time or only at the dimension of features) or all of them together.

Recently, dilated convolutions have been employed for SED, showing promising results. Specifically, [11] presented an approach where dilated convolutions were used together with RNNs for the task of SED, focusing explicitly on the exploitation and learning of long temporal patterns. In [2] presented an adaptation of the CRNN architecture, using DWS and dilated convolutions, using dilation only

1 https://github.com/tensorflow/models/tree/master/research/audioset/yamnet
in the dimension of time (i.e. time-dilated convolution). With the employment of the DWS and time-dilated convolutions, the system in [2] had 85% less parameters than the CRNN, while at the same time, achieved better performance on a typical SED dataset. The improvement achieved with the dilated convolutions indicates that these convolutions can be used for the effective modelling of longer temporal context, similarly to what RNNs do.

A typical process that is applied to models dealing with sequences and using an RNN and a classifier, is the conditioning of each input to the RNN with the immediate previous prediction of the classifier. For example, in an RNN-based sequence-to-sequence model, the input to the first RNN of the decoder can be conditioned with the predictions of the classifier of the model. This conditioning is called language modelling (LM), and has been incorporated extensively in natural language modelling [18, 19, 20] and, recently, in SED [21]. The benefit of LM is that the RNN can be informed on the previous predictions of the classifier, enforcing a contextual modelling of the class activities by the RNN, and leading to the learning of longer intra- and inter-class temporal patterns. For example, specific sound events tend to appear sequentially in time, e.g. “car horn” and “car passing by”. Other sound events can be active in non-consecutive short time frames (e.g. around 40 ms), for example “footsteps”. Using LM for SED, a method can benefit by modelling these class dependencies, resulting in increased performance [21].

In this work we present a novel algorithm and a method for employing a mechanism similar to LM, but for time-dilated convolutions (i.e. dilated convolutions using dilation only at the time dimension). We build upon previous methods for SED, which employ LM [21] and depth-wise separable and time-dilated convolutions [2], and we present a variant of the time-dilated convolutions. Our presented method includes a time-dilated convolution that is conditioned on the predictions of the classifier. To evaluate our method, we employ a freely available SED dataset, and we assess the performance on typically used and frame-wise calculated metrics, i.e. F1 score and error rate (ER). The rest of the paper is organized as follows. In Section 2 we present our method, and we describe the obtained results and their discussion in Section 4 and Section 5 concludes the paper.

2. PROPOSED METHOD

Our proposed method takes as an input a sequence \( \mathbf{X} \in \mathbb{R}^{T \times F} \) of \( T \) vectors, with each vector consisting of \( F \) features, and applies in series a learnable feature extractor, a temporal pattern identifier, and a classifier. The output of the method is another sequence of \( T \) vectors, \( \mathbf{y} = [y_1, \ldots, y_T] \), with \( y_t \in [0, 1]^C \) representing the activity predictions of \( C \) sound event classes for each time step \( T \). We employ a learnable feature extractor consisting of DWS convolutions and a time-dilated convolution, used as the temporal pattern identifier. To take advantage of intra- and inter-class activity patterns, we propose a novel algorithm for conditioning the time-dilated convolution, using the predictions of the classifier, like a language modelling.

2.1. Learnable feature extractor

Our learnable feature extractor consists of \( L \) DWS convolution blocks. The \( l \)-th DWS convolution block gets as an input the output of the previous block, \( \mathbf{H}_{l-1} \in \mathbb{R}^{H_l^{0} \times H_l^{1} \times H_l^{2}} \), where \( H_l^{0} \), \( H_l^{1} \), and \( H_l^{2} \) are the amount of channels, the height, and width of the output of the \( l-1 \)-th DWS convolution block, respectively. \( \mathbf{H}_0 = \mathbf{X} \) is the input to our method with \( H_0^{0} = 1 \), \( H_0^{1} = T \), and \( H_0^{2} = F \). The output of the \( l \)-th DWS block is \( \mathbf{H}_l \in \mathbb{R}^{H_l^{0} \times H_l^{1} \times H_l^{2}} \). Each DWS convolution block consists of a DWS convolution operation, a normalization process, a sub-sampling process, a non-linearity, and a dropout.

The DWS convolution operation itself, consists of two convolutions with unit stride, a leaky rectified linear unit (LReLU), and a normalization process. The first convolution of the DWS convolution operation of the \( l \)-th DWS convolution block, learns spatial information on its input \( \mathbf{H}_{l-1} \), by utilizing \( H_l^{1} \) kernels \( K_l \in \mathbb{R}^{K_l^{0} \times K_l^{1} \times K_l^{2}} \), and is given as

\[
\mathbf{D}_l^{k_l} = \text{BN}((\text{LReLU}(\mathbf{D}_l^{k_l} \mathbf{H}_{l-1}^{k_l} - K_l^{0} \mathbf{H}_{l-1}^{k_l} - K_l^{d})), \text{BN}(1)),
\]

where \( D_l \in \mathbb{R}^{H_l^{1} \times D_l^{1} \times D_l^{2}} \) is the output features of the first convolution at the \( l \)-th DWS convolution block. \( D_l \) is used as an input to a LReLU and a batch-normalization process as

\[
\text{D}_l' = \text{BN}(\text{LReLU}((\text{D}_l))), \text{BN}(1),
\]

where \( \text{LReLU}(\alpha) = \begin{cases} \alpha, & \text{if } \alpha \geq 0, \\ \beta \alpha, & \text{otherwise} \end{cases} \). (3)

BN is the batch-normalization process, \( 0 \leq \beta < 1 \) is a hyper-parameter of the LReLU function, and \( D_l' \in \mathbb{R}^{H_l^{1} \times D_l^{1} \times D_l^{2}} \) is the output of the BN. The second convolution of the DWS convolution operation learns cross-channel information. It utilizes \( H_l^{1} \) kernels, \( z_l \in \mathbb{R}^{H_l^{1} \times d_l^{1}} \), takes as an input \( D_l' \), and processes it as

\[
\mathbf{S}_l^{k_l} = \sum_{k^{l-1} = 1}^{H_l^{1} - 1} \mathbf{D}_l^{k_l} \mathbf{H}_{l-1}^{k^{l-1}} z_l^{k^{l-1}},
\]

where \( S_l \in \mathbb{R}^{H_l^{1} \times D_l^{1} \times D_l^{2}} \) is the output of the second convolution operation at the \( l \)-th DWS convolution operation. The output of the DWS block is obtained as

\[
\mathbf{H}_l = \text{BN}(\text{ReLU}(S_l)),
\]

where \( \text{ReLU} \) is the rectified linear unit. A sub-sampling operation (e.g. max-pooling) over the dimension of features is applied after each BN of a DWS block, and a dropout with probability \( p \), where \( p \) is a hyper-parameter. Appropriate zero-padding is applied by each DWS, so that the length of the sequence in \( H_l \) is the same as the sequence length of \( X \). Finally, \( H_L \) is reshaped to \( H' \in \mathbb{R}^{H_l^{0} \times H_l^{n}} \), where \( H_n = H_{L}^{0} \) and \( H_l^{n} = H_{L}^{1} \cdot H_{L}^{2} \).

2.2. Conditioned time-dilated convolutions

\( H' \) is given as an input to a 2D conditioned time-dilated convolution (CDCNN), which is followed by a linear layer with a softmax activation function that acts as a classifier. A 2D CDCNN is a typical 2D time-dilated convolution (DCNN), which has an extra input channel. This extra input channel is initialized to zeros and, during the forward pass, is populated with jointly-learned embeddings of the predictions of the classifier that follows the DCNN.
The 2D DCNN, with \(K^o\) kernels \(K' \in \mathbb{R}^{K^h \times K^w}\), operates on \(H'\) as

\[
O_{k^h, o^w, o^w} = (K' \ast H')(h^b - \xi^h \cdot K^h, h^w - \xi^w \cdot K^w)
\]

\[
= \sum_{k^h=1}^{K^h} \sum_{k^w=1}^{K^w} H' h^b - \xi^h \cdot K^h, h^w - \xi^w \cdot K^w \end{align}
\]

where \(O \in \mathbb{R}^{C \times O^w \times O^w}\) is the output of the 2D DCNN, and \(\xi^h \in \mathbb{R}^{N^h}\) is the time dilation factor of the kernel \(K'\). The process of a DCNN is illustrated in Figure 1.

Then, \(O\) is reshaped to a matrix to represent a sequence of \(O_h\) vectors with \(O_h \in K^o \times K^o\) features, i.e. \(O\) is transformed to \(O' \in \mathbb{R}^{O_h \times \mathbb{R}^{G(\|K^o\|)}}\). \(O'\) is given as an input to the subsequent classifier \(\text{Cls}(\cdot)\) with shared weights through time, to do SED as

\[
y_{o'_h} = \text{Cls}(O'_{o'_h})
\]

Taking advantage of the cross-channel learning process of the time-dilated convolutions and inspired by language modelling methods for SED, e.g. [21], here we propose a novel algorithm for the conditioning of the time-dilated convolutions, based on Eq. (6).

With our algorithm, we aim to take advantage of the previous predictions of the classifier and make the time-dilated convolution to learn intra- and inter-class patterns, by processing simultaneously its input features and the predictions of the classifier for multiple time-steps (like a language modelling). To do so, we sequentially calculate each \(o'_{o'_h}\). This will allow us to apply Eq. (1) on each \(o'_{o'_h}\), and use the output of the CLs for conditioning the prediction of \(O'_{o'_h+1}\). To overcome any dimensionality mismatching for the conditioning of the prediction of \(O'_{o'_h+1}\), we employ a trainable affine transform with a bias, \(\text{Aff} : [0, 1]^C \mapsto \mathbb{R}^{H'w}\). Our proposed algorithm is given in Algorithm 1. The parameters of our proposed method (including the DWS blocks, the CLs, the Aff, and the CD-CNN) can be jointly optimized using the typical, for SED, cross-entropy loss and \(\hat{Y}\).

3. Evaluation

To evaluate our method, we employ a publicly available SED dataset, the frame-wise F1 score and error-rate (ER) as the metrics, and a published method with DWS and non-conditioned time-dilated convolutions as our baseline. We choose the specific baseline because it has achieved the best frame-wise F1 and ER values at our employed dataset [2], and also employs DWS and (non-conditioned) time-dilated convolutions. The code for both our method and our employed baseline is based on the PyTorch framework and is available online.

3.1. Dataset and data pre-processing

We employ the TUT-SED Synthetic dataset as our dataset. TUT-SED Synthetic can be found online[1] and has been previously used in multiple SED papers [2] [5] [21] [22]. This dataset consists of 100 mixtures, approximately eight seconds long, and split in 60%-20%-20% fashion for training, validation, and testing (respectively). TUT-SED Synthetic contains overlapping, isolated sound events from \(C = 16\) 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”, with a maximum polyphony of 5. From each mixture, we extract non-overlapping sequences of \(T = 1024\) vectors with \(F = 40\) log-scaled mel-bands energies, using a window of \(\approx 22\) ms, with 50% overlap between successive windows, and the Hamming windowing function. We normalize all extracted feature vectors to have zero-mean and unit-variance, based on statistics calculated on the training split. We use the extracted features \(X\) as the input to our method.

\[https://github.com/dr-costas/dslam\]
\[https://github.com/dr-costas/dnd-sed\]
\[http://www.cs.tut.fi/sgn/arg/\]
\[taslp2017-crnn-sed/tut-sed-synthetic-2016\]
3.2. Hyper-parameters, metrics, and evaluation process

For evaluating the impact of our proposed conditioned time-dilated convolution, we follow the exact same evaluation process with the method employing DWS convolution blocks and non-conditioned time-dilated convolutions [2]. We consider the method using non-conditioned time-dilated convolutions as our baseline (denoted as Base), we employ the same hyper-parameters, and we assess the results, the replacement of the non-conditioned time-dilated convolution with a conditioned one.

Specifically, we use \( L = 3 \) DWS convolution blocks, with unit stride, \( K_1^n = K_1^{dw} = 5 \), and a padding of \((2, 2)\). The max-pooling operation after each DWS block has kernels and strides of \( \{1, 5\} \), \( \{1, 2\} \), and \( \{1, 2\} \), and for the dropout we use \( p = 0.25 \). Following our baseline, we employ three different kernel sizes for the CDCNN, namely \( (K_1^n, K_2^n) \) \( \in \{ (3, 3), (5, 5), (7, 7) \} \). We employ a dilation factor of \( \xi \in \{ 1, 10, 50, 100 \} \) and a zero-padding at the time dimension of \( 1 \cdot \xi \) for the \( K_1^n = K_2^n = 3 \), \( 2 \cdot \xi \) for \( K_1^n = K_2^n = 5 \), and \( 3 \cdot \xi \) for \( K_1^n = K_2^n = 7 \). We use this zero padding in order to have the same time resolution between the input and the output of the CDCNN, i.e. \( O^h = H^\xi \).

We optimize our method on the training split of the TUT-SED Synthetic, employing the Adam optimizer and a batch size of 16. For the optimizer, we employ the values of its hyper-parameters (i.e. \( \epsilon, \beta_1 \), and \( \beta_2 \)) that are reported at the corresponding paper [24]. For the LReLU we use \( \beta = 1e - 2 \), because is the default value of \( \beta \) in PyTorch. After each epoch, we measure the loss on the validation split and we stop the optimization process if the validation loss has not improved for 30 consecutive epochs. After the optimization process stops, we utilize the values for the parameters of our method that yielded the lowest loss on the validation split.

For the assessment of the performance of our proposed method, we use the same metrics employed for Base, which are the frame-based \( F_1 \) score (the higher, the better) and frame-based error rate (ER, the lower the better). We assess all the combinations of \( \xi \) and \( (K_1^n, K_2^n) \), conducting ten different experiments with each combination, and we report the average and standard deviation of \( F_1 \) and ER. We denote our method as CDNN and the different values for \( (K_1^n, K_2^n) \) using sub-scripts. For example, CDNN\(_{10,3}\) is the combination of \( \xi = 10 \) and \( K_1^n = K_2^n = 3 \). Accordingly, BASE\(_{100,7}\) denotes the baseline method with the combination of \( \xi = 100 \) and \( K_1^n = K_2^n = 7 \).

Finally, to evaluate solely the addition of the conditioning without using time-dilated convolution, we also assess the performance of our method with \( \xi = 1 \). That is, we still employ the conditioning as described in Algorithm[1] but there is no time-dilation for the CDCNN. This case is denoted with a value of 1 for \( \xi \), e.g. CDNN\(_{1,3}\) for \( \xi = 1 \) and \( K_1^n = K_2^n = 3 \).

### 4. RESULTS AND DISCUSSION

In Table[1] are the obtained results for our method, with all the combinations of \( \xi \) and \( (K_1^n, K_2^n) \). Additionally, in Table[1] we report the difference of the results obtained with our method that uses the conditioning of the time-dilated convolution, and our baseline which is using a non-conditioned time-dilated convolution.

![Table 1: Obtained average (Avg) and STD F1 score and ER for our method, with all the combinations of \( \xi \) and \( (K_1^n, K_2^n) \). \( \Delta_{CDNN-Base} \) is the difference between the results of our method (CDCNN) and Base, for the corresponding average value of \( F_1 \) or ER.](image)

As can be seen from Table[1] using the conditioning of the time-dilated convolution offers a boost mostly when the dilation factor \( \xi \) is \( 1 < \xi < 50 \). This fact reveals that, on one hand, the conditioning of the time-dilated convolution benefits the method mostly when there is a time-dilation, i.e., when \( \xi > 1 \). On the other hand, and taking into account the results for \( \xi = 50 \) and \( \xi = 100 \), it can be seen that increased dilatation and conditioning, tend to impact the method almost negatively. Though, for \( \xi = 10 \) it can be observed that the method constantly yields the best results for each \( (K_1^n, K_2^n) \).

Finally, in Table[1] can be observed that the best results are obtained for \( \xi = 10 \) and CDNN\(_{10,7}\). This is in accordance with the BASE method, where the best results with conditioning of the time-dilated convolutions, are also obtained for \( \xi = 10 \) and CDNN\(_{10,7}\). This fact clearly indicates that the conditioning of the time-dilated convolution can improve the performance. Though, more detailed research is needed to explore the learning of the inter- and intra-class activity patterns that the method probably is learning with the conditioning.

### 5. CONCLUSIONS

In this paper we presented a novel algorithm for the conditioning of the time-dilated convolutions. Our algorithm mimics the language modelling technique for RNNs, by applying a similar conditioning with the previous predictions of a classifier, but on time-dilated convolutions. To evaluate our method, we employed an existing SED method that is based on DWS and time-dilated convolutions, and we applied our algorithm. We carefully designed the evaluation process, so that the only variable that could affect the results is the application of our algorithm. The obtained results showed that with our algorithm, we SED method could gain a 2% increase on the \( F_1 \) score and a 3% decrease on the ER.

Future research direction, could be focused on the impact of our proposed algorithm on the learning of inter- and intra-class activity patterns. This could be done, by investigating the performance with our algorithm, on different sound event classes that exhibit these kind of patterns, for example “footsteps”, “clock ticking”, “cars passing by”.

6. REFERENCES

[1] Y. Li, M. Liu, K. Drossos, and T. Virtanen, “Sound event detection via dilated convolutional recurrent neural networks,” in ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020, pp. 286–290.

[2] K. Drossos, S. I. Mimilakis, S. Gharib, Y. Li, and T. Virtanen, “Sound event detection with depthwise separable and dilated convolutions,” in 2020 International Joint Conference on Neural Networks (IJCNN), Jul. 2020.

[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, 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.

[6] 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.

[7] 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. 3448, Mar. 2019. [Online]. Available: http://dx.doi.org/10.1109/JSTSP.2018.2885636

[8] 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.

[9] 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.

[10] S. Adavanne, P. Pertilä, and T. Virtanen, “Sound event detection using spatial features and convolutional recurrent neural network,” in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Mar. 2017.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] 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. Tchamitchan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 286–297.

[17] 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.

[18] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in Advances in Neural Information Processing Systems 27, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3104–3112. [Online]. Available: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[19] M. Auli, M. Galley, C. Quirk, and G. Zweig, “Joint language and translation modeling with recurrent neural networks,” in Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1044–1054. [Online]. Available: https://www.aclweb.org/anthology/D13-1106

[20] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, “Scheduled sampling for sequence prediction with recurrent neural networks,” in Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, ser. NIPS’15. Cambridge, MA, USA: MIT Press, 2015, pp. 1171–1179. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969239.2969370

[21] 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.

[22] 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), Sep. 2018.

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