CONVOLUTIONAL RECURRENT NEURAL NETWORK FOR AUDIO EVENTS CLASSIFICATION

Technical Report

Federico Colangelo, Federica Battisti, Alessandro Neri, Marco Carli

Department of Engineering, Via Vito Volterra, 62
00146 Rome, Italy
federico.colangelo, federica.battisti, alessandro.neri, marco.carli@uniroma3.it

ABSTRACT

Audio event recognition is becoming a hot topic both in the research and in the industrial field. Nowadays, thanks to the availability of cheap sensors, the acquisition of high-quality audio is much easier. However, new challenges arise: the large number of inputs requires adequate means for coding, transmitting, and storing the recorded data. Moreover, to build systems that can act based on their surroundings (e.g. autonomous cars), automatic tools for detecting specific audio events are needed. In this paper, the effectiveness of an architecture based on a combination of convolutional and recurrent neural networks for general purpose audio event detection is evaluated. Specifically, the architecture is evaluated in the context of the DCASE challenge on general purpose audio tagging, in order to provide a clear comparison with architectures based on different principles.

Index Terms— Audio event recognition, deep neural network, audio classification

1. GENERAL DESIGN PRINCIPLES

The first step in the design of a Deep Neural Networks (DNN)-based audio classifier is the choice of the type of input used for training the system (e.g., time or frequency domain, Mel-Frequency Cepstral Coefficients (MFCC) coefficients, etc.). The purpose of using transformed domains/features is to reduce the dimensionality of the input while retaining as much information as possible, due to the fact that larger input sizes result in more parameters and thus is exponentially more complex optimization problems that must be solved to train the system (i.e., *curse of dimensionality*).

When dealing with audio signal, the frequency domain is usually preferred. As shown in [1], improved performances are obtained by training DNN models in the frequency domain. More specifically, the Short-Time Fourier Transform (STFT) of the input signal is calculated to account for the non-stationary frequency content of audio signals. The STFT is typically computed on overlapping frames, in order to avoid introducing artifacts. Nevertheless, while large overlaps are often used in spectral audio processing (especially when high fidelity is needed), smaller overlaps often result in better classification performances, since the STFT size is reduced and thus the model is trained on smaller inputs. To further reduce the classifier input size, Mel coefficients, computed from the power spectrogram of the input, can be exploited. Mel filters have been extensively used in speech recognition, since they offer better frequency resolutions (i.e. shorter triangle bases) at low frequency values. Despite the loss of frequency information, the simplification of the optimization problem and the improved performances have made Mel coefficients one of the most used approach in audio machine learning.

The temporal dimension of the input also needs to be carefully addressed. Audio events have a strong inter-class variability in the temporal support. Furthermore, publicly available dataset are usually composed of noisy recordings that do not start and stop with the event. To cope with this issue, the input can be padded to match the length of the longest sample. However, this approach becomes unfeasible when the variance of the sample length is high, as the classifier is forced to learn parameters tuned for input of different effective length.

A different approach is to train the classifier with slices extracted from the input sequence, under the hypothesis that the selected slice length contains enough information to discriminate between the classes. Reducing the input size also has a positive effect on the complexity of the optimization problem. Moreover, this approach is better suited for systems operating *in the wild*. In fact, in this way there is no need to perform a detection step before the classification, and the system should be more resilient to cases where only a portion of the input signal is recorded.

The patches extracted from a Mel spectrogram can be used to train a machine learning model. DNNs, both convolutional and recurrent [2], have shown to outperform state-of-the-art methods in classification tasks for audio signals. Each model offers a different advantage: Convolutional Neural Networks (CNN)s have very good performances with high-dimensional data that are invariant to translation.

Specifically, the output of a convolutional layer is obtained by convolving c filters (convolution kernels) with the input, obtaining a feature map with c channels, thus detecting the pattern associated with a single filter independently of their position in the signal. A Mel spectrogram can still be considered invariant to translation in the time dimension. On the other hand, Recurrent Neural Networks (RNN)s are well-suited to deal with sequential data, since long sequences can be processed step-by-step with a limited memory of previous sequence elements.

In this contribution, a Long Short-Term Memory (LSTM) approach is used. LSTM are RNN models optimized to learn long-range patterns by means of the additional parameters controlling the *memory* of the model. The structure of a LSTM unit is shown in Figure 1. Basically, to deal with the vanishing/exploding gradient problem, in a LSTM cell an additional parameter (the cell state $\mathbf{C}_t$) is used. Each unit in the cell generates an output and an update for

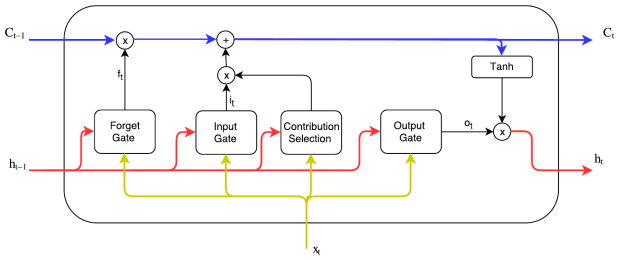

Figure 1: Architecture of an LSTM unit

$\mathbf{C}_t$ based on its current input and on the learned parameters. In the following, $\mathbf{W}, \mathbf{U}, \mathbf{b}$ represent the learned parameters of the LSTM cell, while the subscripts indicate the gate to which they belong. The amount of information that is kept from the previous cell state $\mathbf{C}_{t-1}$ (i.e. past inputs), is modulated by f_t, an integer in the [0 1] range computed as follows:

$$f_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \quad (1)$$

where $\mathbf{x}_t$ is the current input, $\mathbf{h}_{t-1}$ represents the previous unit output, and σ is the sigmoid function.
The part of the current input that is *memorized* in the cell state is computed by the Contribution Selection block and the input gate. More specifically, the Contribution Selection block computes the information to be added to C_t, C_t^{up} by means of:

$$\mathbf{C}_t^{up} = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c),$$

while the input gate computes i_t, an integer in the range [0 1], that is used to modulate the contribution of $\mathbf{C}_t^{up}$ to $\mathbf{C}_t$ and that is calculated as:

$$i_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i), \quad (2)$$

where $\mathbf{C}_t$ is updated by means of Equation 3:

$$\mathbf{C}_t = f_t \mathbf{C}_{t-1} + i_t \mathbf{C}_t^{up}. \quad (3)$$

Finally, the output of the unit, $\mathbf{h}_t$, is calculated with the following equation:

$$\mathbf{h}_t = o_t \circ \tanh(\mathbf{C}_t) \quad (4)$$

where o_t is a scalar in the range [0, 1] given by the Output Gate:

$$o_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o). \quad (5)$$

Since CNN and RNN present complementary advantages, it is possible to combine them, as done in [3], by exploiting convolutional layers as feature extractors and by using the output for training a LSTM. Given this principle, the convolutional layers should be used to learn features in the frequency axis while maintaining unaltered the temporal dimension of the data.

Finally, in the case of many modern machine learning datasets, the dataset can contain data with a noisy ground-truth. In order to deal with this issue, label smoothing is used. Label smoothing is a regularization technique used to prevent a machine learning model from becoming *too confident* in its prediction (i.e. having an output that is sparse). In practice, label smoothing consists in applying the following transformation to the one-hot encoded label, δ_n [4]:

$$y'(n) = (1 - l)\delta_n + \frac{l}{N_c} \quad (6)$$

Type	Patch size	Dilation factor	Output channels
Convolutional	7x7	1	128
Max pooling	2x2	-	-
Convolutional	5x5	2	128
Max pooling	2x1	-	-
Convolutional	5x5	2	256
Max pooling	2x1	-	-
Convolutional	3x3	2	256
Max pooling	2x1	-	-
Convolutional	1x1	1	64
Max pooling	2x1	-	-

Table 1: Parameters of the convolutional layers

where n is the class index, N_c is the number of classes and l is the smoothing factor.

2. ARCHITECTURE DETAILS

The training set is processed by extracting the STFT of each audio sample. A Mel filter-bank, composed of 128 triangular filters, is applied to the power spectrogram computed from the STFT and the logarithm function is applied to the coefficients. The number of filters is selected based on [5, 3]. The model is trained by extracting patches of 128 temporal steps from the Mel spectrogram, with the starting index randomly selected. Audio samples leading to shorter Mel spectrograms are zero-padded to match the minimum time length.

The neural network is composed of a stack of convolutional layer connected to a LSTM. The model has 5 convolutional layers, whose parameters are shown in Table 1. Batch normalization [6] is applied to the output of each layer. Dilated convolutions are used in the middle convolutional layers to exploit the benefits of a larger filter-size while keeping limited the number of parameters [7].
The main peculiarities of the convolutional layers are the asymmetric Max pooling and the reduction of the number of output channels in the last layer. Max pooling is mostly used across the frequency dimension for preserving the temporal structure.
The purpose of the last convolutional layer is to reduce the dimensionality of the LSTM input. To this aim, 1x1 convolutions are used, as suggested in [8]. In this way, the dimensionality is reduced while retaining only the most useful information from the 256 channels.
The final state of the LSTM cell is connected to a feed-forward layer composed of N_c units, N_c being the number of classes in the dataset.
The model is trained by means of the RMSProp algorithm [9], with initial learning rate equal to 0.001. The learning rate is decreased by a factor of 0.5 if there is no improvement in the validation set loss for more than 10 epochs. An initial batch size of 50 is used for training the model. The batch size is then halved after the 100th and 130th epoch. Cross-entropy is used as loss function.
To deal with labels that have not been manually verified (noisy), two smoothing constants are used: a smaller smoothing factor, l_1 for data with reliable ground truth and a larger factor l_2 for data with noisy ground truth. A smoothing factor l_2 of 0.1 is used for samples with noisy ground-truth while the verified labels are not smoothed ($l_1 = 0$).
The model is regularized by applying dropout after each convolutional layer with a probability of keeping the activation of 0,2 and by

adding the L2 norm of the model parameters is to the cost function a factor of 0.002.

3. EXPERIMENTAL TESTS

3.1. Dataset

The proposed model has been trained on the DCASE 2018 task 2 dataset [10]. The dataset contains 18873 audio files taken from the Freesound archive [11], a crowd-sourced audio library. Audio files are encoded as mono PCM audio files, with a sampling frequency of 44.1 kHz. The length of the events is heterogeneous, ranging from below one to more than 30 seconds. Each file belongs to one over 41 classes. The label ontology is a sub-set of the one defined in [12].

The files are divided into a training and a test set. The training set is composed of 9474 audio samples, for which the ground-truth is provided. However, only about one third of the training set ground-truth has been manually verified, while the remaining data is estimated to be annotated with 70% correctness. 1427 samples from the training dataset are used as a validation set to perform hyper-parameters optimization.

3.2. Experimental results

The assignment of a label during the testing phase is performed as follows: the sample is transformed according to the procedure described in the Section 2. Every slice extracted from the same audio event is processed. The label for a single event is given by the sum of the model output over all the slices extracted. The performances of the model are evaluated by means of the accuracy and the Mean Average Precision (MAP)@3 scores. The MAP@N is given by:

$$MAP@N = \frac{1}{N_s} \sum_{n_s}^{N_s} \sum_{k}^{N} P(k)$$

where N_s is the number of evaluated samples and $P(k)$ is the precision at the k_{th} prediction. It is worthful to note that this result is obtained without cross-validation or model averaging.

In the framework of the challenge, the ground-truth of the test set is not available. More specifically, the only result that can be calculated over the test set is the MAP@3 score over 19% of the test set. In the test phase, the model yields an MAP@3 of 73, thus showing improved performances with respect to the baseline system.

The following results are computed over the validation set. The model yields an overall validation accuracy of 77,98% and an MAP@3 score of 82,75 %. Table 2 shows more detailed information over the validation results. As can be noticed, the samples that are predicted with less accuracy belongs to classes that are underrepresented in the dataset, suggesting that a more balanced training set could yield better performances.

Class	**Accuracy (%)**	**Training samples (% of training set)**
Applause	95	3
Oboe	93,87	3.12
Cowbell	90,47	2.12
Snare drum	90,47	3.22
Shatter	89,28	3.05
Double bass	89,13	3.17
Burping/eructation	88,88	2.29
Violin or fiddle	88,88	3.3
Glockenspiel	86,66	0.99
Fart	86,04	3.21
Trumpet	85,45	3.06
Bark	83,33	2.54
Cough	81,08	2.57
Hi-hat	80,76	3.1
Computer keyboard	80	1.30
Finger snapping	80	1.21
Acoustic guitar	79,48	3.26
Microwave oven	79,16	1.52
Saxophone	79,06	3.21
Keys jangling	78,26	1.45
Bass drum	77,77	2.96
Knock	77,77	2.81
Harmonica	77,14	1.62
Laughter	77,08	3.15
Drawer open or close	76,92	1.65
Gong	76,31	3.17
Flute	74,46	3.16
Fireworks	74,28	3.31
Clarinet	73,17	3.24
Cello	72,91	3.15
Tambourine	72,09	2.22
Meow	68	1.62
Chime	66,66	1.25
Telephone	61,53	1.34
Bus	59,09	1.09
Scissors	57,89	0.95
Writing	57,89	2.90
Gunshot/gunfire	56,25	1.64
Tearing	54,54	3.2
Squeak	54,05	3.29
Electric piano	38,46	1.55

Table 2: Per-class accuracies over the validation set correlated with the percentage of per-class training set samples

4. CONCLUSIONS

The task of audio event recognition has gained increased relevance in the last years, thanks to the rapid development of tools based on the paradigm of DNN that ease this task while increasing its accuracy. Many models have been introduced and tested on different datasets, thus picturing a confusing landscape regarding which method performs best. In this paper we presented an approach based on the combination of CNN and RNN. The obtained results shows that a simple architecture outperforms the baseline system based on CNN alone while using a comparably simple approach. In the future, more experiments will be performed to see if the proposed model continues to outperform CNN-based methods even when the pre-processing is optimized.

5. REFERENCES

[1] L. Hertel, H. Phan, and A. Mertins, "Comparing time and frequency domain for audio event recognition using deep learning," *CoRR*, vol. abs/1603.05824, 2016. [Online]. Available: http://arxiv.org/abs/1603.05824

[2] F. Colangelo, F. Battisti, M. Carli, A. Neri, and F. Calabr, "Enhancing audio surveillance with hierarchical recurrent neural networks," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Aug 2017, pp. 1–6.

[3] E. Cakir, 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, June 2017.

[4] 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)*, June 2016, pp. 2818–2826.

[5] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, March 2017.

[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[7] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, vol. abs/1511.07122, 2015. [Online]. Available: http://arxiv.org/abs/1511.07122

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, 2014.

[9] T. Tieleman and G. Hinton, "Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning," https://goo.gl/P9g5tc, Last accessed: July 2018.

[10] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline," 2018, submitted to DCASE2018 Workshop. [Online]. Available: https://arxiv.org/abs/1807.09902

[11] E. Fonseca, J. Pons, X. Favory, F. Font, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra, "Freesound datasets: a platform for the creation of open audio datasets," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, 2017, pp. 486–493.

[12] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP 2017*, 2017.