INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

A Transfer Learning Based Feature Extractor for Polyphonic Sound Event Detection Using Connectionist Temporal Classification

Yun Wang and Florian Metze

Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, U.S.A.
{yunwang, fmetze}@cs.cmu.edu

Abstract

Sound event detection is the task of detecting the type, onset time, and offset time of sound events in audio streams. The mainstream solution is recurrent neural networks (RNNs), which usually predict the probability of each sound event at every time step. Connectionist temporal classification (CTC) has been applied in order to relax the need for exact annotations of onset and offset times; the CTC output layer is expected to generate a peak for each event boundary where the acoustic signal is most salient. However, with limited training data, the CTC network has been found to train slowly, and generalize poorly to new data.

In this paper, we try to introduce knowledge learned from a much larger corpus into the CTC network. We train two variants of SoundNet, a deep convolutional network that takes the audio tracks of videos as input, and tries to approximate the visual information extracted by an image recognition network. A lower part of SoundNet or its variants is then used as a feature extractor for the CTC network to perform sound event detection. We show that the new feature extractor greatly accelerates the convergence of the CTC network, and slightly improves the generalization.

Index Terms: sound event detection (SED), connectionist temporal classification (CTC), transfer learning, convolutional neural networks (CNN)

1. Introduction

Sound event detection (SED) is the task of detecting the type, onset time, and offset time of sound events in audio. The current state of the art uses recurrent neural networks (RNNs) [1, 2, 3, 4]. These networks make a prediction at each time step. For monophonic SED, where only one sound can be active at a given moment, the RNN employs a softmax output layer to generate a distribution over all target events, from which the event with the highest probability is considered active. For polyphonic SED, where multiple sound events can overlap, the RNN dedicates one output neuron to each event and performs binary classification. In either case, the frame-level predictions need to be smoothed to generate a (type, onset, offset) tuple for each occurrence of a sound event.

In order to train these RNNs that make frame-level predictions, it is necessary to annotate the exact onset and offset times of sound events in the training data, which can be a tedious process. Inspired by the successful application of connectionist temporal classification (CTC) [5] to speech recognition, CTC has also been used for SED [6]. CTC is an objective function that computes the total probability of a sequence of input tokens, marginalizing over all possible alignments (*i.e.* onset and offset times of sound events); with CTC, it is sufficient to annotate the training data with sequences of sound events, without exact timing information. For polyphonic SED, since it is difficult to define the order of overlapping sound events, the boundaries (*i.e.* onsets and offsets) of sound events are used as tokens, instead of the events themselves. [6] demonstrates preliminary success of applying CTC to sound event detection, and points out that CTC is especially good at detecting short, transient events, which has been hard for conventional methods. However, the system in [6] converges slowly, and generalizes poorly to unseen data.

A limiting factor for the success of CTC is the lack of labeled training data. Because a CTC-RNN needs to figure out the alignment of the input sequences on itself, it takes more data to train a CTC network than a framewise RNN. Unlike speech recognition, for which hundreds or even thousands of hours of training data is easily available, current annotated corpora for SED (*e.g.* the noiseme corpus [7] and the TUT-SED corpus [8]) hardly exceed 10 hours. This not only impairs the generalizing power of networks, but also limits their depth to 1 or 2 layers, so they cannot enjoy all the benefits of deep learning.

A hopeful means to overcome this limitation is *transfer learning*. The image and video analysis community has produced huge corpora with visual annotations; these have been successfully applied to audio analysis tasks such as acoustic scene classification [9]. In this paper, we attempt to learn better representations of sound signals by transferring knowledge from SoundNet [10]. SoundNet is a deep convolutional network that takes raw waveforms as input, and it is trained to predict the objects and scenes in video streams at certain points. The ground truths of the objects and scenes are produced by image recognition networks such as VGG16 [11] or AlexNet [12]. Even though what can be seen in the video may not always be heard in the audio and *vice versa*, with sufficient training data, the network can still be expected to discover the correlation between the audio and the video. After the network is trained, the activations of an intermediate layer can be considered a representation of the audio suitable for object and scene recognition.

SoundNet is a fully convolutional network, in which the frame rate decreases with each layer. In sound event detection, since we need to predict the onset and offsets of sound events with reasonable precision, we cannot extract features from the higher layers of SoundNet directly. However, the higher layers may contain more abstract representations of the audio signals that are more useful for sound event detection. In order to extract features from these layers with sufficient temporal resolution, we train two variants of SoundNet with the top few layers replaced by fully connected layers or recurrent layers that do not reduce the frame rate. We study how these feature representations affect the SED performance, as well as the speed of convergence when training the CTC-RNN.

This work was supported in part by a gift award from Robert Bosch LLC.
This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by NSF grant number OCI-1053575.

http://dx.doi.org/10.21437/Interspeech.2017-1469

Table 1: *The structure of the original SoundNet*

Layer	input	conv1	pool1	conv2	pool2	conv3	conv4	conv5	pool5	conv6	conv7	conv8 (output)
# feature maps	1	16	16	32	32	64	128	256	256	512	1024	1000 + 401
Filter size		64		32		16	8	4		4	4	4
Activation		relu		relu		relu	relu	relu		relu	relu	softmax
Batch normalization		yes		yes		yes	yes	yes		yes	yes	
Subsampling		2	8	2	8	2	2	2	4	2	2	2
Frame rate (Hz)	22,050	11,025	1,378	689	86	43	21.5	10.8	2.69	1.35	0.67	0.34
Reception Field		2.9 ms	3.5 ms	26 ms	36 ms	0.21 s	0.37 s	0.51 s	0.79 s	1.91 s	4.16 s	8.59 s

Table 2: *The structure of SN-F, with layers above "pool5" replaced by fully connected layers.*

Layer	input	conv1	pool1	conv2	pool2	conv3	conv4	conv5	pool5	fc1	fc2	fc3	output
# feature maps	1	16	16	32	32	64	128	256	256	100	100	100	1000 + 401
Filter size		64		32		16	8	4					
Activation		relu		relu		relu	relu	relu		tanh	tanh	tanh	softmax
Batch normalization		yes		yes		yes	yes	yes					
Subsampling		2	5	2	5	2	2	2	4				Correction: Should be 2
Frame rate (Hz)	16,000	8,000	1,600	800	160	80	40	20	10	10	10	10	10
Reception Field		4.0 ms	4.5 ms	24 ms	29 ms	0.12 s	0.21 s	0.29 s	0.44 s	0.44 s	0.44 s	0.44 s	0.44 s

Correction: Should be 0.34 s

2. Model Structure

2.1. The CTC-RNN for Sound Event Detection

Sound event detection is performed by a simple RNN with a CTC output layer, identical to the network in [6]. The input features are fed into a single bidirectional LSTM layer, with 400 hidden units in each direction and the ReLU non-linearity. The CTC output layer has a vocabulary size of $2n + 1$, where $n = 17$ is the number of sound event types; the output tokens are the onset and offset of each sound event type, plus a "blank" token. An output sequence of the CTC layer can be reduced to a sequence of event boundaries by first collapsing consecutive repeated tokens into a single one, and then removing the blank tokens. The network is trained to maximize the total probability of all output sequences that can be reduced to the ground truth sequence of event boundaries. Best-path decoding is performed during testing, *i.e.* we take the most probable token at each time step, and reduce this sequence of output tokens into a sequence of event boundaries.

2.2. SoundNet and Its Variants for Feature Extraction

The input features for the CTC-RNN are provided by SoundNet [10] or its two variants, SN-F and SN-R.

SoundNet is a fully convolutional network that predicts objects and scenes from raw waveforms. The input is a monaural waveform with a sample rate of 22,050 Hz. The network has seven hidden convolutional layers, interspersed with max-pooling layers. Each convolutional layer doubles the number of feature maps and halves the frame rate; each max-pooling layer halves the frame rate as well. The output layer is also convolutional. It has 1,401 output units, split into two softmax groups of sizes 1,000 and 401, standing for the distributions of objects and scenes, respectively. The structure of SoundNet is summarized in Table 1.

The output layer of SoundNet has a frame rate of about 1/3 Hz. During training, the audio tracks of 20-second video excerpts are fed into the network. This corresponds to about 6.7 frames, but considering boundary effects, the output only contains the distributions of objects and scenes at 4 time steps.

Table 3: *The higher, recurrent layers of SN-R. Layers up to "pool5" are identical to SN-F.*

Layer	gru1	gru2	gru3	output
# feature maps	100×2	100×2	100×2	1000 + 401
Activation	relu	relu	relu	softmax
Batch normalization	yes	yes	yes	
Frame rate (Hz)	10	10	10	10

Ground truth distributions are extracted using VGG16 [11] from the video track at 3 s, 8 s, 13 s, and 18 s. The network is trained to minimize the KL divergence from the ground truth distributions to the predicted distributions. There is a misalignment between the timestamps of the two distributions, but it is ignored.

To localize the onsets and offsets of sound events with reasonable precision, the CTC-RNN for sound event detection must run at a sufficient frame rate. Conventionally, we have set this to 10 Hz [6]. In SoundNet, only one layer ("conv5") has a frame rate close to this value. Therefore, we use the lower part of SoundNet (up to layer "conv5") as a feature extractor for the CTC-RNN.

It may be expected that higher layers of SoundNet compute representations of the input audio that are closer to objects and scenes, *i.e.* closer to sound events. However, these layers in SoundNet have been subsampled too much to be used for SED. In order to make use of the information in the higher layers, we train two variants of SoundNet, SN-F and SN-R. Instead of using convolutional layers all the way up, we switch to *fully connected* (SN-F) or *recurrent* (SN-R) layers after the frame rate has been reduced to the desired value of 10 Hz. After three fully connected or recurrent layers, a fully connected output layer performs the object and scene classification. Also, we have changed the input sampling rate to 16,000 Hz to match the "noiseme" corpus [7] we use for SED. The structures of SN-F and SN-R are summarized in Tables 2 and 3. The values at the "pool5" layer or any higher layer may be used as input features for the CTC-RNN.

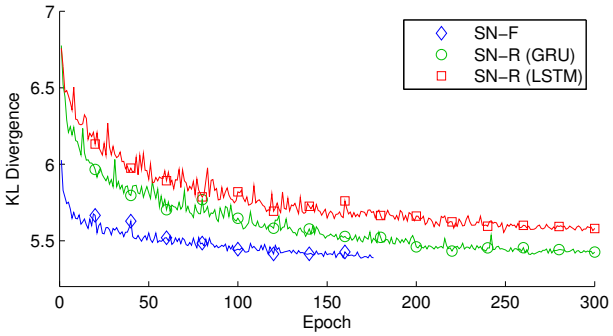

Figure 1: *Training the variants of SoundNet: The evolution of the validation KL divergence of SN-F and SN-R, the latter using either GRU or LSTM cells.*

3. Experiments

3.1. Training the Variants of SoundNet

We train SN-F and SN-R using the same data as SoundNet, which can be downloaded from the SoundNet demo page (`https://projects.csail.mit.edu/soundnet/`). The training data contains the videos from the YFCC100M [13] corpus as well as additional Flickr videos, totaling about 2 million. Truncated to at most 20 seconds long, the total duration of these videos amounts to about 1 year. The number of frames with automatically generated object and scene distributions is about 7 million. The validation set contains data of similar quality, whose size amounts to 1/15 of the training set; we randomly picked 1,000 videos.

We optimized the networks using the Keras [14] toolkit. The loss function was the sum of the KL divergences of the object and scene distributions, measured in nats per frame. We used a batch size of 64 videos (identical to the original SoundNet), and checked the loss on the 1,000-video validation set after every 160 minibatches, which we call an epoch. Each epoch took about 18 minutes; going over the entire training set would take 2.5 days. The original SoundNet was trained using the Adam optimizer with a constant learning rate of 0.001; we decayed the learning rate with a factor of 0.9 when the minimum validation loss did not see any update for 5 epochs. We found this decay helpful for the network to reach a lower loss.

We studied the effect of the recurrent cell type for SN-R, as well as the effect of the activation function. It turned out that GRU cells [15] reached a lower KL divergence than LSTM cells [16], but the activation function did not make a difference for either SN-F or SN-R. In Fig. 1, we plot the evolution of the validation loss of SN-F and SN-R, all using the "tanh" activation function. SN-F converged faster thanks to its simpler structure; by Epoch 175 (after seeing about 90% of the training data), it reached a validation loss of 5.39. We trained SN-R until Epoch 300. With LSTM cells, the final validation loss was 5.58; with GRU cells, 5.43. For comparison, the loss of the original SoundNet on the 1,000-video validation set (3,418 frames) is 5.15, but this number was measured after excluding about 2% of the frames, because on these frames SoundNet predicted zero probabilities for some object or scene classes.

We also studied the effect of batch normalization [17]. The original SoundNet used batch normalization for all the convolutional layers, and we found it essential to do the same. In the fully connected or recurrent layers, batch normalization made no difference on the KL divergence, but we found it to slightly improve the SED performance of SN-R. Consequently, for the experiments in the next subsection, we used a SN-R with GRU cells, the ReLU non-linearity and batch normalization in the recurrent layers, as described in Table 3. Because the non-linearity is not the final step of computation in GRU cells, batch normalization was applied after all the GRU computation. This is different from the convolutional layers, where batch normalization was performed *before* the non-linearity.

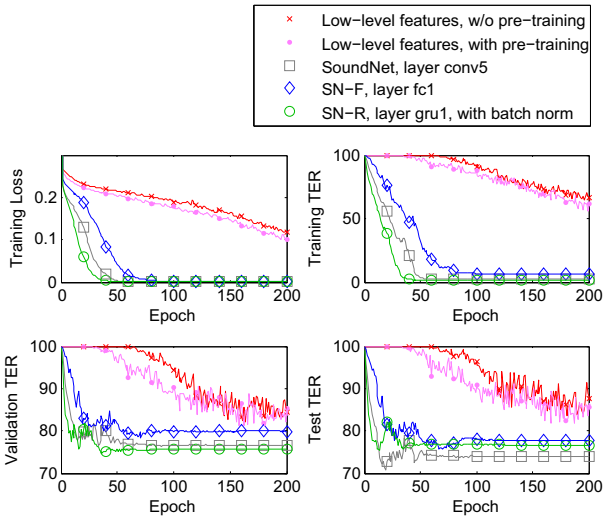

Figure 2: *Training the CTC-RNN for sound event detection: The evolution of the training loss and the token error rate (TER) on the training, validation and test sets, using either low-level acoustic features or transfer learning features extracted from SoundNet or its variants. Note that low-level features do not yet achieve convergence at 200 epochs. Also note the different scales of the training TER vs the validation and test TER, which indicates severe overfitting.*

3.2. Sound Event Detection Using a CTC-RNN

We conducted SED experiments on the noiseme corpus [7], with a setup almost identical to [6]. The corpus contained 464 recordings totaling 9.6 hours, annotated with 17 sound event types. The data was partitioned into training, validation and test sets with a duration ratio of 3:1:1.

We implemented the CTC-RNN using the [18] toolkit. With input features extracted from SoundNet, SN-F or SN-R, pre-training was found to be unnecessary, so we initialized the weight matrices of the CTC-RNN using Glorot uniform initialization [19], and set the initial bias of the forget gates to one [20, 21]. Each minibatch contained 5 sequences of 500 frames each, and an epoch was defined as a pass through all the training data. The loss function was per-frame negative log-likelihood, with the alignment hinting tolerance (see [6] for details) set to 10 frames (*i.e.* each peak was allowed to occur within a 2-second window around the ground truth). We ran the stochastic gradient descent (SGD) algorithm for 200 epochs with a Nesterov momentum [22] of 0.9. The learning rate was initialized to 3.0, and was decayed by a factor of 0.8 when the token error rate on the validation set saw no update in 5 epochs. The token error rate (TER) is computed the same way as word error rate (WER), treating sound event boundaries as words.

Fig. 2 shows the evolution of the loss function and the TER on the training, validation and test sets, using the "conv5" layer

Table 4: *Evaluating the CTC-RNN: Token error rate (TER) at convergence (Epoch 200) using features extracted from different layers of SoundNet, SN-F and SN-R. "BN" means after batch normalization. The TER values of low-level features are measured at Epoch 500.*

Feature	Layer	#Dims	Train TER	Val. TER	Test TER
Low-level	N/A	50	15.2	82.8	81.0
SoundNet	conv5	256	2.3	**76.6**	**74.0**
SN-F	pool5	256	3.5	80.5	77.4
	fc1	100	6.1	79.9	77.8
	fc2	100	6.5	82.2	79.2
	fc3	100	4.6	80.8	77.6
SN-R	pool5	256	3.0	78.9	**74.9**
	gru1	200	3.0	77.8	78.4
	gru1-BN	200	1.2	**75.8**	76.7
	gru2	200	3.0	84.5	80.6
	gru2-BN	200	2.3	82.3	79.0
	gru3	200	90.6	96.4	96.4
	gru3-BN	200	60.9	90.5	91.2

of SoundNet, "fc1" layer of SN-F, and "gru1" layer of SN-R (after batch normalization), respectively. For comparison, the curves produced using a 50-dimensional low-level acoustic feature [6] are also included. Using features learnt by transferring from an image recognition task substantially accelerated the convergence. When using low-level features, the CTC network exhibited a "warm-up" stage in which it did not output anything; pre-training the network with a frame-wise sound event classifier shortened this stage from 60 epochs to 40 epochs. But with transferred features, the warm-up stage was almost non-existent. The final test-set TER was also lower than using low-level features (see Table 4). Actually, before we switched to SoundNet features, we had tried several techniques on the CTC-RNN (including dropout [23] and data augmentation) in order to improve the generalization, but none of these techniques brought the test TER below 80%. The transfer learning based features broke this barrier easily; however, the gap between the training and test sets remained huge.

Next, we look at which layer of SoundNet or its variants yielded features that led to the best SED performance. Table 4 shows the TER on the training, validation and test sets after 200 epochs when using features extracted from different layers. We find that features extracted from the "conv5" layer of the original SoundNet remains competitive. With SN-F, features extracted from the higher, fully connected layers yield better SED performance than low-level features, but still fall short of SoundNet's "conv5" layer. With SN-R, we first notice that it is always better to extract features after batch normalization. We also notice the SED performance gets worse as features are extracted from higher layers, which is counter-intuitive.

We try to understand the cause of this performance deterioration by visualizing the activations of some higher layers of SN-F and SN-R in Fig. 3. We can see a clear transition in the activations at 4.5 s, which is preserved in all the layers of SN-F. In SN-R, however, the transition becomes blurred out at the "gru2" layer, and disappears altogether at the "gru3" layer. This indicates that recurrent layers, which have access to information at distant moments, may not be good at representing local information. The fully connected layers of SN-F, on the other hand, maintain a reception field of 0.44 seconds, and therefore are able to concentrate on what happens within this time window.

Correction: Should be 0.34 seconds

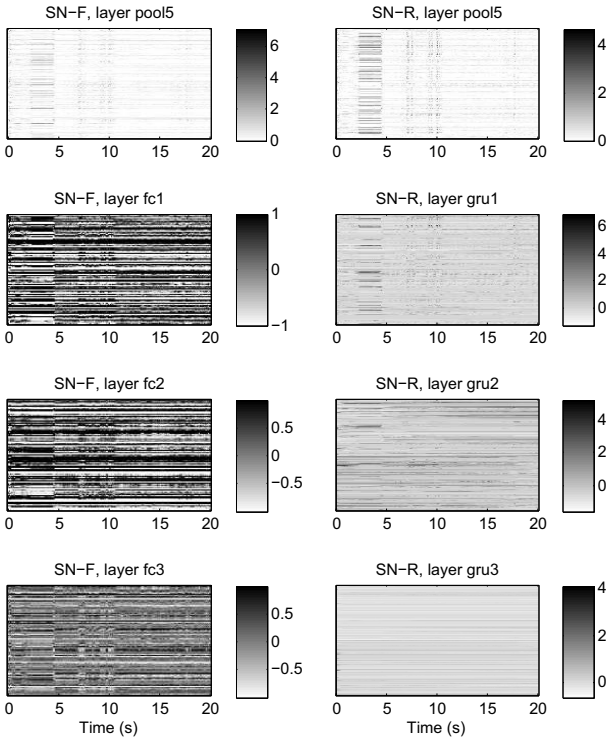

Figure 3: *The activations of the higher layers of SN-F and SN-R on a validation recording. For the recurrent layers of SN-R, the activations have been batch normalized.*

4. Conclusion and Future Work

Sound event detection with a CTC-RNN suffers from a lack of labeled training data. In this paper, we have studied the possibility of transferring knowledge learnt from an image recognition task to help with sound event detection. We extracted features from intermediate layers of SoundNet and its two variants, SN-F and SN-R, to replace the low-level acoustic features used in the past. The new features greatly accelerated the training of the CTC-RNN for sound event detection, and slightly improved its generalization.

We expected that features extracted from layers closer to the target would yield better SED performance, but we were not able to observe this in the experiments. With SN-R, the reason is the loss of temporal resolution in the recurrent layers. SN-F, which maintained its temporal resolution by limiting the size of its reception fields, was also unable to close the gap between the training and testing token error rates. This indicates the necessity of a careful analysis of the errors made by the SED network, in order to find out where the bottleneck of the performance lies.

Because the training data for SoundNet is labeled for visual objects and scenes, and the labels are generated automatically, there may be a limit to what can be learnt from this data for sound event detection. Recently, Google released Audio Set [24], a huge manually annotated corpus for SED. It contains 2 million 10-second audio excerpts taken from YouTube videos, weakly labeled with the presence or absence of 632 sound event types. Since Audio Set is directly annotated for sound event detection, it can be expected that features learnt from this corpus may lend further assistance to CTC-based SED. We will explore this possibility in the future.

5. References

[1] Y. Wang, L. Neves, and F. Metze, "Audio-Based Multimedia Event Detection Using Deep Recurrent Neural Networks", in *Proc. of ICASSP*, 2016.

[2] G. Parascandolo, H. Huttunen, and T. Virtanen, "Recurrent neural networks for polyphonic sound event detection in real life recordings", in *Proc. of ICASSP*, 2016.

[3] S. Adavanne, *et al.*, "Sound event detection in multichannel audio using spatial and harmonic features", in *Workshop on Detection and Classification of Acoustic Scenes and Events*, 2016.

[4] T. Hayashi, *et al.*, "Bidirectional LSTM-HMM hybrid system for polyphonic sound event detection", in *Workshop on Detection and Classification of Acoustic Scenes and Events*, 2016.

[5] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks", in *Proc. of ICML*, 2006.

[6] Y. Wang and F. Metze, "A first attempt at polyphonic sound event detection using connectionist temporal classification", in *Proc. of ICASSP*, 2017.

[7] S. Burger, Q. Jin, P. F. Schulam, and F. Metze, "Noisemes: manual annotation of environmental noise in audio streams", technical report CMU-LTI-12-07, Carnegie Mellon University, 2012.

[8] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection", in *Proc. of EUSIPCO*, 2016.

[9] S. Mun, *et al.*, "Deep neural network based learning and transferring mid-level audio features for acoustic scene classification", in *Proc. of ICASSP*, 2017.

[10] Y. Aytar, C. Vondrick, and A. Torralba, "SoundNet: Learning sound representations from unlabeled video", in *Proc. of NIPS*, 2016.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", in *Proc. of ICLR*, 2015.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks", in *Proc. of NIPS*, 2012.

[13] B. Thomee, *et al.*, "YFCC100M: The new data in multimedia research", in *Communications of the ACM*, vol. 59, no. 2, 2016.

[14] F. Chollet, "Keras: Deep learning library for Theano and TensorFlow", `https://github.com/fchollet/keras`, 2015.

[15] J. Chung, *et al.*, "Empirical evaluation of gated recurrent neural networks on sequence modeling", in *NIPS 2014 Workshop on Deep Learning*, 2014.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory", in *Neural Computation*, vol. 9, 1997.

[17] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", in *Proc. of ICML*, 2015.

[18] J. Bergstra, *et al.*, "Theano: a CPU and GPU math expression compiler", in *Proc. of the 9th Python for Scientific Computing Conference*, 2010.

[19] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", in *Proc. of the 13th International Conference on Artificial Intelligence and Statistics*, 2010.

[20] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM", in *Neural Computation*, vol. 12, no. 10, 2000.

[21] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures", in *Proc. of ICML*, 2015.

[22] Y. Nesterov, "A method of solving a convex programming problem with convergence rate O(1/sqr(k))", in *Soviet Mathematics Doklady*, vol. 27, 1983.

[23] N. Srivastava, *et al.*, "Dropout: a simple way to prevent neural networks from overfitting", in *Journal of Machine Learning Research*, vol. 15, 2014.

[24] J. F. Gemmeke, *et al.*, "Audio Set: An ontology and human-labeled dataset for audio events", in *Proc. of ICASSP*, 2017.