AUDIO-BASED MULTIMEDIA EVENT DETECTION USING DEEP RECURRENT NEURAL NETWORKS

Yun Wang, Leonardo Neves, Florian Metze

Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, U.S.A.

{yunwang, lneves, fmetze}@cs.cmu.edu

ABSTRACT

Multimedia event detection (MED) is the task of detecting given events (*e.g.* birthday party, making a sandwich) in a large collection of video clips. While visual features and automatic speech recognition typically provide the best features for this task, non-speech audio can also contribute useful information, such as crowds cheering, engine noises, or animal sounds.

MED is typically formulated as a two-stage process: the first stage generates clip-level feature representations, often by aggregating frame-level features; the second stage performs binary or multi-class classification to decide whether a given event occurs in a video clip. Both stages are usually performed "statically", *i.e.* using only local temporal information, or bag-of-words models.

In this paper, we introduce longer-range temporal information with deep recurrent neural networks (RNNs) for both stages. We classify each audio frame among a set of semantic units called "noisemes"; the sequence of frame-level confidence distributions is used as a variable-length clip-level representation. Such confidence vector sequences are then fed into long short-term memory (LSTM) networks for clip-level classification. We observe improvements in both frame-level and clip-level performance compared to SVM and feed-forward neural network baselines.

Index Terms— Multimedia event detection (MED), noisemes, recurrent neural networks (RNNs), long short-term memory (LSTM)

1. INTRODUCTION

With the pervasion of cell-phones in daily life and the popularity of video-sharing websites such as YouTube, recent years have seen an explosive increase in the number of consumer-produced videos. However, the means of retrieving videos in such large collections are still mostly limited to text-based search in human-generated video metadata or voice captions, rather than the actual content of a video. Multimedia event detection (MED) aims to solve this problem, mostly by fusing information from multiple complementary channels.

While the visual content of video clips (*e.g.* people, animals, objects, scenes, actions, and OCR text) often contains the most useful information for event detection, the audio track can also provide important clues. The words that people say may directly reveal the event being shown in a clip. Besides speech (and speech meta-data such as emotions, speaker identities, or discourse information), non-speech sounds often contain additional and complementary audio information: some sounds are directly related to events, such as the cheering of the audience in a soccer game; other sounds indicate the environment in which events happen, such as the buzz of machines in a factory. Some of this audio information may not even be present in the visual content.

MED systems usually consist of two stages. In the first stage, a clip-level representation is generated for each video clip. Such representations usually come in the form of a vector or a sequence of vectors, and are often an aggregation of frame-level features. In the second stage, binary or multi-class classifiers are built for the target events, taking the clip-level representations as inputs.

Two approaches have been shown to be the most successful in audio-based MED. The first approach is inspired by speaker identification techniques. The frame-level acoustic features (such as MFCCs) of a video clip are modeled by a Gaussian mixture model (GMM). The mean vectors and the diagonals of the covariance matrices of a GMM are concatenated to yield a "supervector" [1], which is used as the clip-level representation, such as in [2]. Optionally, "i-vectors" [3] maybe extracted from the supervectors in order to capture most of the variability in a smaller number of dimensions, such as in [4]. The supervectors or i-vectors may be fed into any general-purpose classifier to perform clip-level classification. The second approach is inspired by topic classification in natural language processing. A "vocabulary" is learned by quantizing the frame-level acoustic features, and each video clip is represented by a bag-of-audio-words (BoAW) vector [6]. These vectors may either be classified by general-purpose classifiers, or modeled with latent Dirichlet allocation (LDA) [7], treating each event as a topic.

In both the i-vector and the BoAW approaches, temporal information is not fully exploited: the frame-level feature extraction only makes use of local context, and the clip-level representation ignores the order of the frames altogether. Various ways have been suggested to compensate for this loss. Instead of building clip-level representations upon the features of single frames, researchers have proposed short, semantically meaningful audio segments as the units of building clip-level representations. These units may be either learnt in an unsupervised fashion [8, 9] or defined by humans [10]. The semantic units are usually modeled with hidden Markov models (HMMs), which exploits the temporal information within a unit. In order to make use of temporal information across units, people have included bigram and long-distance co-occurrence counts in the clip-level representation [8], or incorporated a language model of the semantic units during event classification [9].

As deep neural networks (DNNs) are gaining popularity in recent years, researchers have started to apply them to the task of multimedia event detection as well. Since neural networks are mostly used for supervised classification, studies of MED using

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by NSF grant number OCI-1053575.

This work was partially funded by Facebook, Inc. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Facebook, Inc.

The authors would like to thank Yipei Wang for the help in setting up experiments on the TRECVID 2011 MED corpus.

neural networks have mainly focused on frame-level classification of human-defined semantic units. In [11], a deep neural network was used to distinguish between four semantic units; a larger repository of 61 semantic units were dealt with in [12] and [13]. In [14] and [15], deep neural networks were used not only for frame-level classification but also for clip-level detection. While deep neural networks are displaying their strong power in the classification tasks, the networks used in the works above all had a feed-forward structure without temporal recurrency, and therefore did not make full use of temporal information, such as the repetitive structure of many sounds, or typical sequences of sounds.

In this paper, we try to combine the classification power of deep neural networks and the temporal information by using deep recurrent neural networks (RNNs). We adopt a set of semantic units that have interpretable and descriptive meanings, called "noisemes" and first introduced in [10], and classify each frame among 17 noisemes with a deep bidirectional recurrent neural network (BRNN). The sequence of noiseme confidence vectors is directly used as the clip-level representation without pooling it into a single vector, and a deep recurrent neural network with long short-term memory (LSTM) cells is used to make clip-level predictions. In both stages, the recurrent neural networks outperform the baseline of feed-forward neural networks.

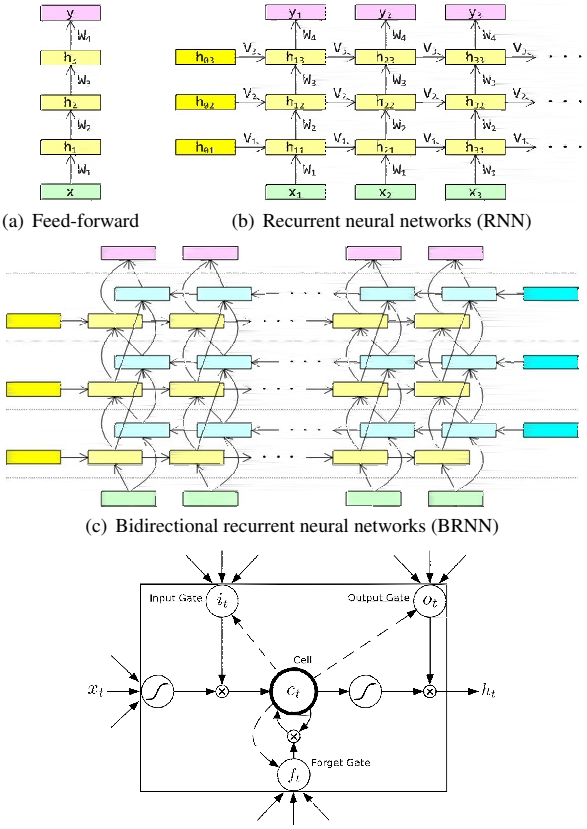
2. DEEP RNNS AND LSTM NETWORKS

Deep neural networks (DNNs) are powerful general-purpose models for classification and regression. The simplest networks have a feed-forward structure, as shown in Fig. 1(a). Each block denotes a layer, which computes a vector. The hidden layers and the output layer multiply their input vector with a weight matrix, and apply a non-linear activation function to produce an output vector. Typical activation functions include sigmoid ($y = 1/(1 + \mathrm{e}^{-x})$), tanh ($y = tanh(x)$), and ReLU ($y = \max(x, 0)$). It is this non-linearity that gives neural networks their incredible learning power. The weight matrices are the parameters of the network. They can be trained with the gradient descent method (or, more commonly, stochastic gradient descent with minibatches), and the gradients can be calculated with the error back-propagation algorithm [16].

Feed-forward networks require a fixed-size input, and cannot deal with sequential data of variable length. Recurrent neural networks (RNNs, Fig. 1(b)) were invented to solve this problem. The same feed-forward structure is copied for each time step, and the hidden layers of adjacent time steps are also connected with weight matrices that share the same value across time steps. The prediction at any time step can now make use of information in the entire history. RNNs can also be trained with gradient descent; the algorithm to compute the gradients is called back-propagation through time (BPTT) [17], and is essentially the same as back-propagation for feed-forward networks.

Sometimes it is desirable to use information not only from the past but also from the future, and this is when bidirectional recurrent neural networks (BRNNs) [18] are useful. In a BRNN, each hidden layer consists of a forward chain and a backward chain going in opposite directions (see Fig. 1(c)). The input and output layers are connected to both chains; both chains in each hidden layer are connected to both chains in the next hidden layer. With this structure, information can flow freely in both directions in the network.

Even though in theory (bidirectional) RNNs can make use of information in the distant past (and future), in practice this is hard to achieve due to a problem called vanishing or exploding gradient [19]. This happens because when the error signal is back-propagated



(a) Feed-forward (b) Recurrent neural networks (RNN)

(c) Bidirectional recurrent neural networks (BRNN)

(d) LSTM cell (peephole connections, indicated by dashed arrows, are not used in this paper)

Fig. 1. *The structures of deep feed-forward networks, deep recurrent networks, deep bidirectional recurrent networks, and the LSTM cell.*

through the network, it gets multiplied with the weight matrices as many times as there are time steps, and amplified or attenuated depending on the magnitude of the weights. To avoid this problem, the simple non-linearity blocks of RNNs can be replaces by long short-term memory (LSTM) cells [20], whose structure is shown in Fig. 1(d). Each memory cell interacts with the surroundings via a set of gates, and their content can be retained for a long time. The memory cells of adjacent time steps are connected with a constant weight of 1, so the error signal does not explode or vanish during back-propagation. LSTM networks have been shown to perform well on tasks where long-range memories are necessary. LSTM networks can also be made bidirectional.

Implementing the error back-propagation algorithm can be complicated, especially for LSTM networks. Luckily, the symbolic derivation functionality of the Theano toolkit [21] takes care of it. The toolkit also provides transparent access to graphical processing units (GPUs), which can speed up the training of neural networks significantly with parallel computation. Most of our experiments were run on NVIDIA K20 and K80 GPUs.

In the experiments, we describe the size of neural networks as $h \times n$, *e.g.* 100×2. The number n indicates the number of hidden layers; the number h indicates the size (*i.e.* vector length) of the hidden layer at each time step and in each chain (in the case of bidirectional networks).

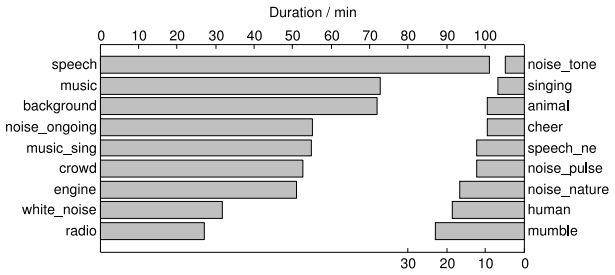


Fig. 2. *Duration of each noiseme in the corpus. "speech_ne" means "non-English speech".*

3. FRAME-LEVEL NOISEME CLASSIFICATION

Our first step toward event detection was to classify audio frames into semantic units called "noisemes". We conducted frame-level noiseme classification on the "noiseme" corpus [10], which consists of 388 video clips, totaling 7.9 hours. The original annotation contains 48 noiseme classes; we manually merged some rare and semantically close classes (*e.g.* "laughing", "crying", "screaming" and "child noises" were merged into "human noises"), and ended up with 17 noisemes (plus a "background" class). The duration of each noiseme in the corpus is shown in Fig. 2. Note that the total length of the bars in Fig. 2 (10.5 h) is longer than 7.9 hours; this is because nearly a third of the duration is labeled with more than one noiseme.

The corpus was divided into training / validation / test sets with a duration ratio of 3:1:1. This was not a trivial task, since we needed to make sure that the duration of each noiseme in the three sets also formed a ratio of 3:1:1. We tried to find such a partitioning with a stochastic algorithm: starting from a random partitioning, we iteratively tried moving a random video clip to a different set, until no such moves could bring the ratio closer to 3:1:1. With 1000 random starts, we were able to find a partition such that for each noiseme, the percentage of duration contained in the three sets deviated from 60%, 20%, and 20% by less than 1%.

We extracted acoustic features using the OpenSMILE toolkit [22]. We first extracted low-level features such as MFCCs and F_0 (fundamental frequency), and then computed a variety of statistics over these raw features using sliding windows of 2 seconds moving 100 ms at a time. This procedure yielded feature vectors with 6,669 dimensions; we selected 983 dimensions using the information gain criterion. This is the same acoustic feature as used in [23].

As a baseline, we built a deep feed-forward neural network to perform 18-way classification. The hidden layers used the ReLU activation; the output layer contained 18 nodes in a softmax group. We used cross entropy ($L = -\sum_{i=1}^{N} \log p(y_i)$) as the objective function. To accommodate frames with multiple labels, the predicted probability of the target class $p(y_i)$ was replaced by the total predicted probability of all the target classes. We chose the frame accuracy as the evaluation metric; a frame was considered as correctly classified as long as one of the target classes had the highest predicted probability.

We trained the network using the stochastic gradient descent algorithm with the Nesterov accelerated gradient [24]. The batch size was 2,500 frames. We adopted an adaptive learning rate schedule: when the validation error rate decreases by more than 1% relative in an epoch, the learning rate is increased by 5%; when the validation error rate decreases by less than 0.5% relative, the learning rate is decreased by 20%; training is terminated when the validation error rate decreases by less than 0.1% relative in 5 consecutive epochs. Regularization and dropout [25] were not used, as we did not find them helpful for this task. For each network configuration, we ran the training four times from different random initializations.

Type	Size	# Params	Init LR	Test Accuracy (%)				Ave.(%)
DNN	500×2	0.75M	0.005	**45.5**	45.2	45.2	44.7	45.1
RNN	500×1	0.75M	0.002	45.3	**48.0**	45.7	46.2	46.3
BRNN	300×2	1.32M	0.002	46.0	47.2	**48.1**	46.9	**47.0**
LSTM	300×1	1.55M	0.005	44.7	46.5	46.3	**47.6**	46.3
BLSTM	300×1	3.09M	0.005	47.0	**48.2**	46.8	45.0	46.7

Table 1. *The optimal configuration and test frame accuracy of various types of neural networks (LR: learning rate; optimal momentum coefficient was 0.9 for all network types).*

We obtained the best test frame accuracy of 45.5% with a 500×2 feed-forward network, an initial learning rate of 0.005, and a momentum coefficient of 0.9. The network had 0.75M parameters. For comparison, a support vector machine (SVM) classifier using the linear kernel achieved a test frame accuracy of 41.5%.

Next, we trained unidirectional and bidirectional recurrent neural networks with either plain ReLU units or LSTM units. We broke all the training clips down to sequences no longer than 500 frames, putting the cutting points in the middle of contiguous "background" frames as much as possible. Each minibatch consisted of 5 sequences, which is comparable to the batch size of 2,500 frames for the feed-forward network. The hyper-parameters of the optimal configuration of each network type can be found in Table 1.

Table 1 also lists the test accuracies of the four runs as well the average test accuracy for each type of neural network. From the average test accuracy, we can see that introducing recurrency produced 1% of improvement, and that introducing temporal memory on both sides yielded another 1%. LSTM cells did not perform better than plain ReLU units; this is probably because classifying a frame does not require memory from a long distance away.

4. CLIP-LEVEL EVENT DETECTION

Frame-level noiseme classification provides a preliminary understanding of the content of video clips, and forms a basis for clip-level event detection. We conducted clip-level event detection on the development data of the TRECVID 2011 Multimedia Event Detection (MED) evaluation [26], which was also used in [2, 7, 23, 27]. The corpus consists of 3,104 training video clips and 6,642 test video clips. A few of the clips do not contain an audio track; we excluded them from our experiment. Some of the videos are labeled with one out of 15 events, listed in Table 2; others are "background" videos. The test set contains more background videos than the training set. We reserved one quarter of the training clips for validation, ensuring that the proportion of each event in the validation set and the rest training set remained the same.

We extracted the same 983-dimensional acoustic feature as in Sec. 3. Then, using the bidirectional RNN that achieved a 48.1% frame accuracy, we transformed the acoustic features into 18-dimensional "noiseme confidence vectors". The sequence of such vectors was used as the clip-level representation of video clips.

As a baseline, we built one feed-forward neural network for each event type to perform clip-level binary classification. Since feed-forward neural networks require a fixed-size input, we took the mean of the noiseme confidence vectors across all the frames. Note that all temporal information is lost in this operation! Hidden layers of the neural network used the ReLU activation, and the output layer consisted of a single sigmoid node to yield a confidence

Correction:
The names of these two events should be swapped.

ID	Event	Train%	Test%
E001	Attempting a board trick	5.23	1.71
E002	Feeding an animal	5.23	1.71
E003	Landing a fish	3.92	1.30
E004	Working on a woodworking project	4.10	1.33
E005	Wedding ceremony	4.58	1.53
E006	Birthday party	2.88	1.33
E007	Changing a vehicle tire	1.83	0.83
E008	Flash mob gathering	2.88	1.33
E009	Getting a vehicle unstuck	2.09	1.02
E010	Grooming an animal	2.22	1.07
E011	Making a sandwich	2.01	0.96
E012	Parade	2.22	1.02
E013	Parkour	1.83	0.80
E014	Repairing an appliance	2.05	0.94
E015	Working on a sewing project	1.96	0.88

Table 2. *The 15 events used in the MED experiment, and their proportion in the training and test sets.*

score between 0 and 1 for the event. Cross entropy was still used as the objective function; average precision (AP) was adopted as the evaluation metric. The average precision evaluates the quality of a ranking. Sort the test instances by their confidence scores in descending order, and suppose the i-th positive instance is ranked at the r_i-th position (we broke ties by always ranking negative instances higher). The average precision is defined as

$$\text{AP} = \frac{1}{n}\sum_{i=1}^{n}\frac{i}{r_i} \tag{1}$$

where n is the total number of positive test instances. The mean of the AP across all events is called the mean average precision (MAP).

The feed-forward network was trained using gradient descent; since there were only 2,294 training instances, these were not divided into minibatches. The same adaptive learning rate schedule was used as in Sec. 3. An optimal MAP of 4.0% was obtained with a 200×2 feed-forward network, an initial learning rate of 0.3, and a momentum coefficient of 0.9. This MAP value falls behind the numbers reported in [23], but it is still a lot better than random guessing (in which case the MAP will be the average percentage of positive test instances across the events, which is 1.2%).

Clip-level event detection is a task where long-term memory may be desirable: the network needs to remember the order in which noisemes occur, but the noisemes themselves and/or the gaps between them may last for an indefinite duration. We trained a deep LSTM network with the same hyper-parameters as the baseline feed-forward neural network. A difference from the LSTM network we trained in Sec. 3 was that target labels were only provided at the last frame of each sequence.

In this procedure we encountered a number of difficulties. The first difficulty was the wide range of sequence lengths: the shortest sequence had only 37 frames, while the longest one had more than 20,000 frames. This not only meant too much data to process, but also made it hard to form minibatches. We shortened all sequences whose length l was greater than 50 to $m = 50\sqrt{l/50}$ frames, by dividing the original sequence evenly into m segments and taking the average within each segment. This operation loses some local temporal information for long sequences, but keeps the long-range temporal information.

Training the LSTM network directly with these shortened sequences did not beat the feed-forward baseline. We suspect that this was because it was hard for the LSTM to find a good local optimum with as little supervision as one frame per sequence. We

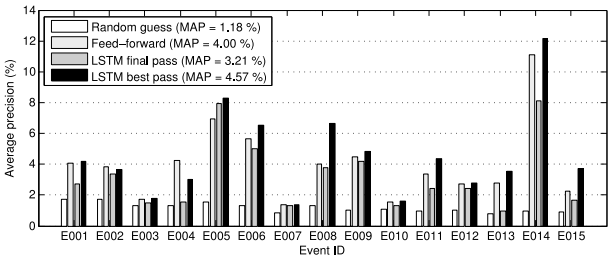


Fig. 3. *The per-event average precision of various networks types.*

devised a multi-pass training strategy to lead the LSTM along a good path. In the first pass, we shortened all the sequences to 1 frame and trained the LSTM network with them, which was equivalent to training a feed-forward network. In each subsequent pass, we trained the LSTM network with sequences shortened to twice the length in the previous pass, starting with a learning rate that was $\alpha = 5$ times the learning rate when the last pass converged. Each pass can be considered as a pre-training for the next pass. After the pass of training with sequences of length 512, we finally fine-tuned the network with the original sequences of variable lengths m.

We observed that the average precision improved for most events during the multi-pass training (especially when the sequence length was around 32), but in the last few passes, the AP fell back to the baseline value or even lower. A closer inspection revealed that this happened because the learning rate multiplier $\alpha = 5$ was too large for the last few passes, and caused the training to diverge. Yet this large multiplier was necessary for the initial passes to encourage the network to explore better regions in the parameter space. This indicates that a more intelligent way of controlling the learning rate is desirable.

If we stopped the multi-pass training after the pass that reached the highest test AP for each event, we did observe an improved MAP of 4.6%, as shown in Fig. 3. This confirms that LSTM networks are able to exploit temporal information for event detection. We believe that the potential of LSTM networks can be brought out to a fuller extent if the learning rate was controlled better.

5. CONCLUSION AND FUTURE WORK

We have demonstrated that deep recurrent neural networks (RNNs) can make use of temporal information to help both the frame-level and clip-level classification stages of multimedia event detection (MED). Using a deep bidirectional recurrent neural network (BRNN), we achieved an improvement of 1.9% absolute in the frame accuracy of noiseme classification; using a deep LSTM network, we improved the mean average precision (MAP) of event detection by 0.6% absolute. The second improvement is still limited by the strategy of controlling the learning rate during network training.

There remains a lot of work to do for both frame-level noiseme classification and clip-level event detection. The noiseme corpus, with 7.9 hours of audio, does not contain enough data to train a reliable classifier for all the 48 types of labeled noisemes. We plan to augment the corpus in a semi-supervised fashion, by iteratively predicting noisemes on more data and incorporating the relatively confident parts into the training set. For event detection, we need to develop a more robust strategy to control the learning rate, so that LSTM networks can fully benefit from the temporal information contained in the sequences.

6. REFERENCES

[1] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, "Support vector machines using GMM supervectors for speaker verification", in *IEEE Signal Processing Letters*, vol. 13, no. 5, pp. 308-311, 2006.

[2] Q. Jin, P. F. Schulam, S. Rawat, S. Burger, D. Ding, and F. Metze, "Event-based video retrieval using audio", in *Proc. of Interspeech*, pp. 2085-2088, 2012.

[3] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification", in *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788-798, 2011.

[4] X. Zhuang, S. Tsakalidis, S. Wu, P. Natarajan, R. Prasad, and P. Natarajan, "Compact audio representation for event detection in consumer media", in *Proc. of Interspeech*, pp. 2089-2092, 2012.

[5] B. Elizalde, H. Lei, and G. Friedland, "An i-vector representation of acoustic environments for audio-based video event detection on user generated content", in *IEEE International Symposium on Multimedia*, pp. 114-117, 2013.

[6] S. Pancoast and M. Akbacak, "Bag-of-audio-words approach for multimedia event classification", in *Proc. of Interspeech*, pp. 2105-2108, 2012.

[7] S. Rawat, P. F. Schulam, S. Burger, D. Ding, Y. Wang, and F. Metze, "Robust audio-codebooks for large-scale event detection in consumer videos", in *Proc. of Interspeech*, pp. 2929-2933, 2013.

[8] B. Byun, I. Kim, S. M. Siniscalchi, and C.-H. Lee, "Consumer-level multimedia event detection through unsupervised audio signal modeling", in *Proc. of Interspeech*, pp. 2081-2084, 2012.

[9] S. Chaudhuri, M. Harvilla, and B. Raj, "Unsupervised learning of acoustic unit descriptors for audio content representation and classification", in *Proc. of Interspeech*, pp. 2265-2268, 2011.

[10] 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.

[11] Z. Kons and O. Toledo-Ronen, "Audio event classification using deep neural networks", in *Proc. of InterSpeech*, pp. 1482-1486, 2013.

[12] O. Gencoglu, T. Virtanen, and H. Huttunen, "Recognition of acoustic events using deep neural networks", in *Proc. of the 22nd European Signal Processing Conference*, pp. 506-510, 2014.

[13] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, "Polyphonic sound event detection using multi-label deep neural networks", in *IEEE International Joint Conference on Neural Networks*, 2015.

[14] M. Ravanelli, B. Elizalde, K. Ni, and G. Friedland, "Audio concept classification with hierarchical deep neural networks", in *Proc. of the 22nd European Signal Processing Conference*, pp. 606-610, 2014.

[15] K. Ashraf, B. Elizalde, F. Iandola, M. Moskewicz, J. Bernd, G. Friedland, and K. Keutzer, "Audio-based multimedia event detection with DNNs and sparse sampling", in *Proc. of the 5th ACM International Conference on Multimedia Retrieval*, pp. 611-614, 2015.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors", in *Nature*, vol. 323, pp. 533-536, 1986.

[17] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model", in *Neural Networks*, vol. 1, pp. 339-356, 1988.

[18] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks", in *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673-2681, 1997.

[19] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies", in *A Field Guide to Dynamical Recurrent Neural Networks*, IEEE Press, 2001.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory", in *Neural Computation*, vol. 9, pp. 1735-1780, 1997.

[21] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. "Theano: a CPU and GPU math expression compiler", in *Proc. of the 9th Python for Scientific Computing Conference*, pp. 1-7, 2010.

[22] F. Eyben, F. Weninger, F. Gross, and B. Schuller, "Recent developments in OpenSMILE, the Munich open-source multimedia feature extractor", in *Proc. of ACM Multimedia*, pp. 835-838, 2013.

[23] Y. Wang, S. Rawat, and F. Metze, "Exploring audio semantic concepts for event-based video retrieval", in *Proc. of ICASSP*, pp. 1360-1364, 2014.

[24] Y. Nesterov, "A method of solving a convex programming problem with convergence rate O(1/sqr(k))", in *Soviet Mathematics Doklady*, vol. 27, pp. 372C376, 1983.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting", in *Journal of Machine Learning Research*, vol. 15, pp. 1929-1958, 2014.

[26] NIST, "2011 TRECVID multimedia event detection evaluation plan", Online: `http://www.nist.gov/itl/iad/mig/upload/MED11-EvalPlan-V03-20110801a.pdf`

[27] Y. Yan, Y. Yang, D. Meng, G. Liu, W. Tong, A. G. Hauptmann, and N. Sebe, "Event oriented dictionary learning for complex event detection", in *IEEE Transactions on Image Processing*, vol. 24, no. 6, pp. 1867-1878, 2015.