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

Virtual Adversarial Training and Data Augmentation for Acoustic Event Detection with Gated Recurrent Neural Networks

Matthias Zöhrer and Franz Pernkopf

Intelligent Systems Group, Signal Processing and Speech Communication Laboratory, Graz University of Technology, Austria

matthias.zoehrer@tugraz.at, pernkopf@tugraz.at

Abstract

In this paper, we use gated recurrent neural networks (GRNNs) for efficiently detecting environmental events of the IEEE Detection and Classification of Acoustic Scenes and Events challenge (DCASE2016). For this acoustic event detection task data is limited. Therefore, we propose data augmentation such as *on-the-fly* shuffling and virtual adversarial training for regularization of the GRNNs. Both improve the performance using GRNNs. We obtain a segment-based error rate of 0.59 and an F-score of 58.6%.

Index Terms: Acoustic event detection, gated recurrent networks, data augmentation, virtual adversarial training

1. Introduction

In order to correctly label acoustic events many different features and models have been suggested in a recent acoustic scene classification challenge summarized in [1]. One of the most popular baseline models are Gaussian mixture models (GMMs) [2] and hidden Markov models (HMMs) [3, 4] using mel-frequency cepstral coefficients (MFCCs). Interestingly, various deep architectures have not been applied in [1]. Recent work however, shows that deep neural networks (DNNs) [5, 6, 7, 8] boost the accuracy when applied to audio data. Due to this fact, they have been explored in the recent IEEE *Detection and Classification of Acoustic Scenes and Events challenge* (DCASE2016) [9]. Especially, convolutional NNs received lots of attention for scene classification [10].

In this paper, we systematically analyze recurrent neural network architectures for acoustic event detection. In particular, we evaluate popular temporal neural network architectures, such as simple recurrent neural networks (RNNs), long-short term memory networks (LSTMs), and gated recurrent neural networks (GRNNs) [11, 12, 13] on environmental audio events of the DCASE2016 data [9]. GRNNs proof themselves in practice through fast and stable convergence rates. By combining GRNNs and virtual adversarial training (VAT) [14] using data of the *real event detection challenge*, i.e. task 3 of DCASE2016, we obtain a segment-based error rate (ER) of 0.59 and an F-score of 58.6%. Unfortunately, data in this task is scarce. Therefore, we additionally introduce data augmentation by *on-the-fly* shuffling.

Main contributions and results are:

- We explore the effects of virtual adversarial training (VAT) to regularize GRNNs.
- We introduce a *on-the-fly* shuffling scheme for data augmentation outperforming the baseline system.
- We focus on gated RNNs outperforming LSTMs and RNNs.

The paper is structured as follows: In Section 2 we shortly introduce the acoustic event detection task and the processing framework including relevant literature. In Section 3 recurrent neural network architectures such as RNNs, LSTMs and GRNNs are presented. Furthermore, we introduce virtual adversarial training (VAT) as regularization technique for GRNNs and an extension of *on-the-fly* data augmentation. Finally, we show experimental results in Section 4. Section 5 concludes the paper.

2. Acoustic Environment Detection

In acoustic event detection different acoustic events embedded in environmental sounds are identified. The environment can be specified based on physical or social context, e.g. park, office, meeting, etc.. Usually, it is differentiated between *polyphonic* and *monophonic* event scenarios: In the first case multiple events can occur at the same time, whereas in the second case no overlapping events exist.

Figure 1 shows the processing pipeline of our acoustic event detection framework. We extract a sequence of feature frames

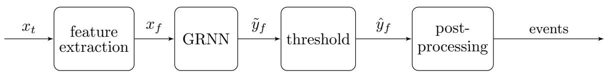

Figure 1: *GRNN event detection system.*

x_f, where $x_f \in \mathbb{R}^D$ and f is the frame index. In particular, we derive *MFCCs* or *log-magnitude spectrograms*, given the raw audio data x_t. These feature frames are processed by a GRNN and class labels are determined by applying individual thresholds on the real-valued output $\tilde{y}_f$ of the multi-label GRNN[1]. This leads to a binary value for each event class at frame-level. Similar as in [9], we post-process the events by detecting contiguous regions neglecting events smaller than 0.1 seconds as well as ignoring consecutive events with a gap smaller than 0.1 seconds.

Recent work on DNNs show a significant boost in classification performance when applied to acoustic event detection. Gencoglu et al. [15] proposed a DNN architecture for acoustic event detection. DNNs are powerful network architectures, but they neglect to model temporal context explicitly. To account for temporal structure, LSTMs have been applied to acoustic keyword spotting [16] and polyphonic sound event detection [17]. LSTMs have a relatively high model complexity and parameter tuning for LSTMs is not always simple.

Most systems of the DCASE 2016 challenge [10] use various kinds of DNNs for event detection. The winning sys-

[1]Note that the thresholds have been determined using the training data.

http://dx.doi.org/10.21437/Interspeech.2017-1238

tem [18] uses several features such as log mel-band energy, harmonic and time difference of arrival features in RNN-LSTM models. Here, we advocate GRNNs [12, 13] for acoustic event classification. GRNNs are a temporal deep neural network with reduced computational complexity compared to LSTMs.

3. Recurrent Neural Network Architectures

We focus on a classical *vanilla* RNN and two popular architectures, the LSTMs and GRNNs.

3.1. Recurrent Neural Networks

Figure 2 shows the flow-graph of an RNN. Given an input x_f^l at layer l the sum between the dot product $W_i^l x_f^l$ and the projected previous hidden state $W_h^l h_{f-1}^l$ is computed, where W_i^l is an input weight matrix and W_h^l the hidden weight matrix, respectively. Next, a non-linear function g is applied. The result is forwarded to the output as shown in Eqn. (1). The bias term b^l is omitted for simplicity.

$$h_f^l = g(W_i^l x_f^l + W_h^l h_{f-1}^l) \tag{1}$$

RNNs are trained via back-propagation through time using a differentiable cost function.

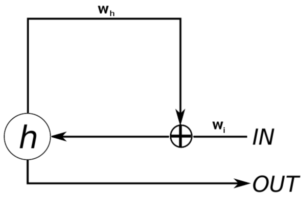

Figure 2: *Flow graph of one vanilla RNN layer.*

3.2. Long Short Term Memory Networks

LSTMs [19, 20] are temporal recurrent neural networks using memory cells to store temporal information. In contrast to RNNs which compute a weighted sum of x_f^l and h_{f-1}^l, each LSTM layer maintains memory cells c^l. Every memory cell stores or erases its content by using *input-* or *reset* gates, represented as internal states i^l and r^l of layer l. An additional *output* gate o^l acts as a weight to access this information. Figure 3 shows the flow-graph of this complex architecture.

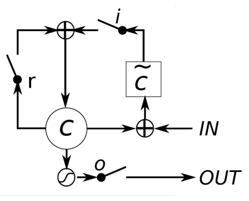

Figure 3: *Flow graph of one LSTM layer [12].*

Mathematically LSTMs are described by Equations (2-7). After feeding the input x_f^l into the model, an input state i_f^l is obtained by applying a sigmoid function σ to the sum of the dot-product of the input weight matrix W_i^l and x_f^l, the projected previous hidden state $U_i^l h_{f-1}^l$ and the projected previous memory cell's content $V_i^l c_{f-1}^l$ of layer l (cf. Equation 2). The reset state r_f^l (cf. Equation 3) and output state o_f^l (cf. Equation 4) are computed in a similar fashion, except for using individual *reset* matrices W_r^l, U_r^l, V_r^l and *output* matrices W_o^l, U_o^l, V_o^l, respectively. The matrices $V \in \{V_i^l, V_r^l, V_o^l\}$ are diagonal. A new memory state, i.e. $\tilde{c}_f^l$, is obtained by applying a *tanh* activation function on the sum of the projected input $W_c^l x_f^l$ and previous hidden memory state $U_c^l h_{f-1}^l$ in Equation (5). After the new memory state values are determined, the memory cell state c_f^l is updated by the previous memory state c_{f-1}^l and $\tilde{c}_f^l$ (cf. Equation (6)), using the reset state r_f^l and the input state i_f^l as weights, respectively. The output as hidden activation states h_f^l is computed by the current memory state $\tanh(c_f^l)$ and the output state o_f^l in Equation (7). Bias variables are omitted for simplicity.

$$i_f^l = \sigma(W_i^l x_f^l + U_i^l h_{f-1}^l + V_i^l c_{f-1}^l) \tag{2}$$

$$r_f^l = \sigma(W_r^l x_f^l + U_r^l h_{f-1}^l + V_r^l c_{f-1}^l) \tag{3}$$

$$o_f^l = \sigma(W_o^l x_f^l + U_o^l h_{f-1}^l + V_o^l c_f^l) \tag{4}$$

$$\tilde{c}_f^l = \tanh(W_c^l x_f^l + U_c^l h_{f-1}^l) \tag{5}$$

$$c_f^l = r_f^l c_{f-1}^l + i_f^l \tilde{c}_f^l \tag{6}$$

$$h_f^l = o_f^l \cdot \tanh(c_f^l) \tag{7}$$

Unlike to the classical RNN, which overwrites the hidden activation at each time-step (cf. Equation 1), LSTMs are able to decide whether to keep or erase existing information with the help of their gates. Intuitively, if LSTMs detect important features from an input sequence at early stage, they easily carry this information over a long distance, hence, capturing potential long-distance dependencies.

3.3. Gated Recurrent Neural Networks

GRNNs are recurrent neural networks (RNNs) using blocks of gated recurrent units. They are a simplification of LSTMs, reaching comparable performance, but having fewer parameters. GRNNs only use *reset-* and *update*-gates. These switches couple static and temporal information allowing the network to learn temporal information. In particular, the *update*-gate z decides to re-new the current state of the model, whenever an important event is happening, i.e. some relevant information is fed into the model at step f. The *reset*-gate r is able to delete the current state of the model, allowing the network to forget the previously computed information. Figure 4 shows the corresponding flow diagram of a GRNN. It gives a visual interpretation how the *update-* and *reset*-gates, i.e. z and r, govern the information in the network. Equations (8-11) describe the network behavior. Starting at the output state h_f^l of layer l, the network uses the *update*-state z_f^l to compute a linear interpolation between past state h_{f-1}^l and current information $\tilde{h}_f^l$ in Equation (8). In particular, the *update*-state z_f^l decides how much the unit updates its content. z_f^l is computed as sigmoid function

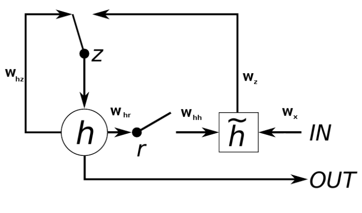

Figure 4: *Flow graph of one GRNN layer [12].*

of the weighted input x_f^l and past hidden state h_{f-1}^l in (9). The W and b denote the weights and bias terms in the model.

$$h_f^l = (1 - z_f^l)h_{f-1}^l + z_f^l\tilde{h}_f^l \tag{8}$$

$$z_f^l = \sigma(W_z^l x_f^l + W_{hz}^l h_{f-1}^l + b_z^l) \tag{9}$$

$$\tilde{h}_f^l = g(W_x^l x_f^l + W_{hh}^l (r_f^l \cdot h_{f-1}^l) + b_h^l) \tag{10}$$

$$r_f^l = \sigma(W_r^l x_f^l + W_{hr}^l h_{f-1}^l + b_r^l) \tag{11}$$

The state $\tilde{h}_f^l$ of the network is computed by applying a non-linear function g to the affine transformed input and previous hidden state h_{f-1}^l in (10). This is similar to *vanilla* RNNs. However, an additional *reset*-state, i.e. r_f^l, is introduced in GRNNs. In particular, an element-wise multiplication is applied between r_f^l and h_{f-1}^l. In Equation (11), the reset state is computed based on the current input frame x_f and the provided hidden state h_{f-1}^l. Multiple GRNN layers can be stacked, forming a deep neural network.

3.4. Virtual Adversarial Training for Regularization

We use virtual adversarial training (VAT) [14, 21] to make the models robust against adversarial perturbations [22, 23]. VAT promotes local smoothness of the posterior distribution $p(y_f|x_f)$ with respect to x_f. In particular, the posterior distribution, i.e. the softmax activation of the network output h_f^l, should vary minimally for small, bounded perturbations of the input x_f. The adversarial perturbation δ_f is determined on frame-level by maximizing the Kullback-Leibler divergence KL-divergence $(\cdot||\cdot)$ of the posterior distribution for unperturbed and perturbed inputs, i.e.

$$\delta_f = \underset{||\delta||<\epsilon}{\mathrm{argmax}}\mathrm{KL}(p(y|x_f)||p(y|x_f+\delta), \tag{12}$$

where $\epsilon > 0$ limits the maximum perturbation, i.e. the noisy input $x_f + \delta$ lies within a radius ϵ around x_f. The smaller the $\mathrm{KL}(p(y|x_f)||p(y|x_f + \delta_f)$ the smoother is the posterior distribution around x_f. Instead of maximizing the conditional likelihood $p(y_f|x_f)$ of the model during training, we maximize the regularized objective

$$\sum_f \log p(y_f|x_f) - \lambda \sum_f \mathrm{KL}(p(y|x_f)||p(y|x_f+\delta_f)), \tag{13}$$

where the tradeoff parameter λ and the radius ϵ have to be selected on development data. VAT can be used for semi-supervised learning. In this case the right part of (13) is applied to unlabeled data samples. This is possible as $\mathrm{KL}(p(y|x_f)||p(y|x_f + \delta_f))$ does not depend on the ground truth labels y_f.

3.5. Data Augmentation

In order to guarantee a stable stochastic optimization of the NNs in the case of limited available data, the data need to be randomized (shuffled). We use a variant of *on-the-fly* shuffling proposed in [24]. Furthermore, we extended this *on-the-fly* shuffling scheme in two directions: (i) we additionally removed frames, *i.e. frame skipping*, from sequence batches every iteration; and (ii) we randomly permuted frames within a window, *i.e. frame shuffling*. Figure 5 gives a visual interpretation of the proposed shuffling. This increases the amount of data by introducing slight permutations and variations of the training sequences.

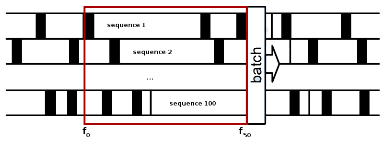

Figure 5: *Extended on-the-fly shuffling of acoustic scene recordings. Black squares indicate skipped frames.*

4. Experiments

4.1. Data

The DCASE2016 sound event detection dataset (task 3) is divided into a training and a test set containing 22 recordings. The dataset has two scene categories, i.e. *home* and *residential area*. These acoustic scenes were selected to represent common environments of interest in applications for (i) safety and surveillance (outside home) and (ii) human activity monitoring or home surveillance. The *home* training corpus contains 11 event classes with 906 events, whereas the *residential area* training corpus contains 7 event classes including 559 events. Table 1 summarizes the events and the number of instances for each scene category, respectively.

Table 1: *Event categories of the DCASE2016 real life audio event detection task.*

Home		Residential Area	
event class	instances	event class	instances
(object) rustling	60	(object) banging	23
(object) snapping	57	bird singing	271
cupboard	40	car passing by	108
cutlery	76	children shouting	31
dishes	151	people speaking	52
drawer	51	people walking	44
glass jingling	36	wind blowing	30
people walking	54		
object impact	250		
washing dishes	84		
water tap running	47		

For evaluation, a segment-based F-score (F) and error-rate (ER) is calculated using segments of one second. The F-score is $F = \frac{2PR}{P+R}$, where P and R are precision and recall. The ER is the normalized sum over all substitutions S, deletions D, and insertions I, i.e. $ER = \frac{\sum S(k)+\sum D(k)+\sum I(k)}{\sum N(k)}$, where N is the number of events in each segment k. More details about the data and the evaluation setup are provided in [9]. 4-fold cross-validation was used to evaluate the models, i.e., for training we use 3-folds and the remaining fold is used for validation. We report the average classification accuracy for all 4-folds of this dataset. The labels of the evaluation set are not published yet.

4.2. Experimental Setup

We pre-processed all recordings with a STFT using a Hamming window with window-size 40ms and 50% overlap. Next, MFCCs including Δ- and Δ^2-features were computed. All features were normalized to zero-mean unit variance using the training corpus. For the experiments we used either MFCCs + Δ + Δ^2 features, resulting in a 60-bin vector per frame as in [9] or raw 1025-bin log magnitude spectrograms. The best model configuration has been determined by grid search, i.e. GRNNs using 3 layers, 200 neurons per layer, sigmoid output layer (+softmax for training the VAT objective) and rectifier activations turned out to be best. Overlapping frame labels of polyphonic events have been removed in the training corpus, forcing the model to extract event class specific features. Each model was trained using the MSE of the predictions of the network output activations on frame-level.

4.3. Results

4.3.1. Comparison of GRNN input features

Table 2 lists the segment-based error (ER) and F-score of GRNNs with MFCC and *log-magnitude* spectrograms. The use of spectrogram features slightly improves the classification accuracy in GRNNs. Therefore, we provide results for RNNs and LSTMs using only the spectrogram features. Overall a 3-layer GRNN achieved an segment-based ER of 0.76 and an segment-based F-score of 40.9%. The baseline is a GMM with 15 components trained with MLE.

Table 2: *Classification results with a 3-layer GRNN using a MSE objective and MFCC or log-magnitude spectrograms.*

Model	Features	Objective	Segment-based ER	F [%]
baseline	MFCC	MLE	0.91	23.7
GRNN	MFCC	MSE	0.76	35.9
RNN	spectrogram	MSE	0.85	26.1
LSTM	spectrogram	MSE	0.84	25.8
GRNN	spectrogram	MSE	**0.76**	**40.9**

4.3.2. Data Augmentation and VAT

Table 3 shows the results of GRNNs using the extended *on-the-fly* data augmentation method in Section 3.5. We processed 100 randomly selected batches, cropped to a fixed length of 50 frames for each optimization step. Furthermore, additional Gaussian noise ($\sigma = 1$) has been added to the input signal x_f. By doing so, we ensure proper randomization of the data for stochastic optimization. Furthermore, the *frame skipping* probability of $p = 0.5$ and uniform *index permutation* within a window of size $w = 3$ was evaluated for a 3-layer GRNN using raw *log-magnitude* spectrograms. The parameters w and p have been determined using grid-search. Both, augmentation methods improve the ER and F-scores. In particular, the best setup achieved a segment-based ER of 0.67 and a segment-based F-score of 51.7%, respectively.

Table 3: *Data augmentation for a 3-layer GRNN and the MSE objective using log-magnitude spectrogram features.*

Model	flipping	shuffling	VAT	Segment-based ER	F [%]
GRNN	-	-	-	0.76	40.9
GRNN	$p = 0.5$	-	-	0.70	48.4
GRNN	$p = 0.5$	$w = 3$	-	**0.67**	**51.7**
GRNN	$p = 0.5$	$w = 3$	VAT	**0.59**	**58.6**

Furthermore, Table 3 shows the results using a VAT regularized GRNN in addition to *on-the-fly* shuffling with $p = 0.5$ and $w = 3$. The VAT configuration of $\lambda = 0.1, \epsilon = 0.1$, and $I_p = 1$ has been determined by grid search. VAT regulatization is able to slightly improve the results, i.e. VAT regularized GRNNs trained on log-magnitude spectrograms achieved an overall ER-score of 0.59% and F-score of 58.6%.

For these performances, Table 4 lists the detailed segment-based ER and F-scores for the scene categories *home* and *residential area*. Environmental sounds in *home* areas, are not classified very well, whereas events in *residential areas* are better recognized.

Table 4: *Classification results for scene categories.*

Model	Acoustic Scene	Segment-based ER	F [%]
GRNN	home	0.71	45.9%
GRNN	residential area	0.48	71.3%
GRNN	average	0.59	58.6%

Table 5 shows segment-based performance results for the individual categories using a GRNN without data augmentation trained on spectrogram data (see Table 2). The error rate for some of the categories is poor. This is often the case for event classes with a low number of instances.

Table 5: *Detailed segment-based ER and F scores using a GRNN without data augmentation.*

Home Event	F [%]	ER	Residential Area Event	F [%]	ER
(object) rustling	7.9	1.79	(object) banging	0.0	1.00
(object) snapping	0.0	1.00	bird singing	64.6	0.60
cupboard	0.0	1.00	car passing by	73.7	0.48
cutlery	8.3	0.96	children shouting	4.6	2.43
dishes	47.1	0.88	people speaking	52.5	0.69
drawer	0.0	1.00	people walking	0.0	1.00
glass jingling	0.0	1.00	wind blowing	16.6	0.91
object impact	0.0	1.00			
people walking	0.0	1.00			
washing dishes	32.7	0.88			
water tap running	58.5	0.73			

Table 6 shows segment-based performance results for the individual categories using a VAT regularized GRNN for comparison. Large ER scores indicate hypersensitivity against specific sound categories, i.e. many insertions occur. Nevertheless, there are improvements due the use of data-shuffling and VAT.

Table 6: *Detailed segment-based ER and F scores using a VAT regularized GRNN.*

Home Event	F [%]	ER	Residential Area Event	F [%]	ER
(object) rustling	58.9	1.00	(object) banging	22.2	1.29
(object) snapping	0.0	1.00	bird singing	80.5	0.38
cupboard	0.0	1.00	car passing by	79.1	0.38
cutlery	75.5	0.40	children shouting	22.6	2.15
dishes	51.6	0.76	people speaking	75.4	0.41
drawer	0.0	1.00	people walking	29.5	1.14
glass jingling	0.0	1.00	wind blowing	63.	0.54
object impact	39.0	0.82			
people walking	31.7	0.94			
washing dishes	44.3	0.73			
water tap running	58.6	0.69			

5. Conclusion

We applied gated recurrent neural networks (GRNNs) for acoustic event detection using the IEEE DCASE2016 challenge data for acoustic event detection (task 3). In this task data is scarce. Therefore, regularization techniques such as virtual adversarial training are applied. Furthermore, we use data augmentation and extend *on-the-fly* shuffling. Both approaches improve the performance. In particular, we obtain a segment-based ER-score of 0.59 and a segment-based F-score of 58.6%.

6. Acknowledgements

This work was supported by the Austrian Science Fund (FWF) under the project number P27803-N15 and I2706-N31. Furthermore, we acknowledge NVIDIA for providing GPU computing resources.

7. References

[1] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Trans. Multimedia*, vol. 17, no. 10, pp. 1733–1746, 2015.

[2] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *European Signal Processing Conference (EUSIPCO)*, 2010, pp. 1267–1271.

[3] J. Keshet and S. Bengio, "Discriminative keyword spotting," in *Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods*. Wiley Publishing, 2009, pp. 173–194.

[4] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 4087–4091.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*, 2014, pp. 3104–3112.

[6] A. Graves, N. Jaitly, and A.-R. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013, pp. 273–278.

[7] M. Zöhrer and F. Pernkopf, "General stochastic networks for classification," in *Advances in Neural Information Processing Systems 27*, 2014, pp. 2015–2023.

[8] M. Zöhrer, R. Peharz, and F. Pernkopf, "Representation learning for single-channel source separation and bandwidth extension," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 12, pp. 2398–2409, 2015.

[9] A. Mesaros, T. Heittola, and T. Virtanen, "Tut database for acoustic scene classification and sound event detection," in *European Signal Processing Conference (EUSIPCO)*, Budapest, Hungary, 2016.

[10] DCASE 2016, "http://www.cs.tut.fi/sgn/arg/dcase2016/," in *Web*, 2016.

[11] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *CoRR*, vol. abs/1409.1259, 2014.

[12] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[13] ——, "Gated feedback recurrent neural networks," *CoRR*, vol. abs/1502.02367, 2015.

[14] T. Miyato, S. Shin-ichi Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional smoothing by virtual adversarial examples." *CoRR*, vol. abs/1507.00677, 2015.

[15] O. Gencoglu, T. Virtanen, and H. Huttunen, "Recognition of acoustic events using deep neural networks," in *European Signal Processing Conference (EUSIPCO)*, 2014, pp. 506–510.

[16] G. Chen, C. Parada, and T. N. Sainath, "Query-by-example keyword spotting using long short-term memory networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5236–5240.

[17] G. Parascandolo, H. Huttunen, and T. Virtanen, "Recurrent neural networks for polyphonic sound event detection in real life recordings," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 6440–6444.

[18] S. Adavanne, G. Parascandolo, P. Pertil, T. Heittola, and T. Virtanen, "Sound event detection in multichannel audio using spatial and harmonic features," DCASE2016 Challenge, Tech. Rep., September 2016.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013, inproceedings, pp. 6645–6649.

[21] M. Ratajczak, S. Tschiatschek, and F. Pernkopf, "Virtual adversarial training applied to neural higher-order factors for phone classification," in *Interspeech*, 2016.

[22] A. Makhzani, J. Shlens, N. Jaitly, and I. J. Goodfellow, "Adversarial autoencoders," *CoRR*, vol. abs/1511.05644, 2015.

[23] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[24] G. Heigold, E. McDermott, V. Vanhoucke, A. Senior, and M. Bacchiani, "Asynchronous stochastic optimization for sequence training of deep neural networks," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.