Recurrent neural networks for polyphonic sound event detection in real life recordings

Giambattista Parascandolo, Heikki Huttunen, Tuomas Virtanen

Tampere University of Technology

giambattista.parascandolo@tut.fi

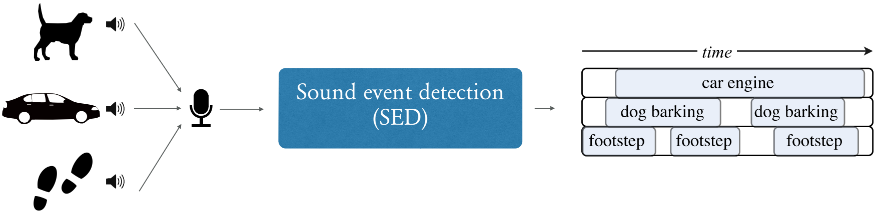
March 24, 2016

Overview

1. Introduction
2. Method
3. Evaluation

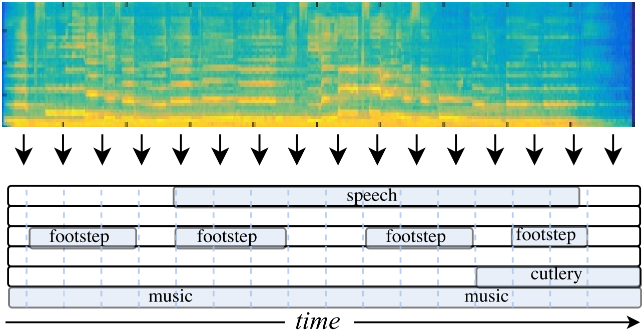
Polyphonic sound event detection in real life recordings

Goal: detect **which** sources are active, **beginning** and **ending** moments.



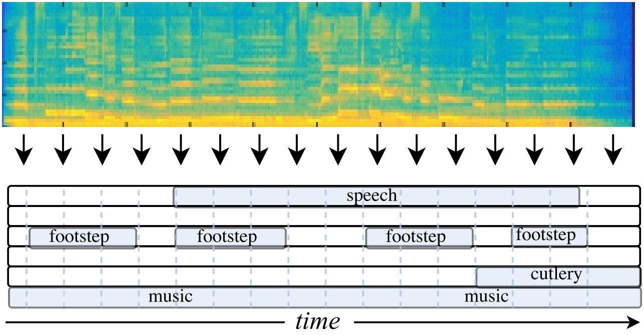
Polyphonic sound event detection

- A multilabel classification task.
- Map input signal to class labels in short time windows (~50ms)



Polyphonic sound event detection

- A multilabel classification task.
- Map input signal to class labels in short time windows (∼50ms)



Previous work

Context dependent

- GMM+HMM
- Nonnegative matrix factorisation (NMF)

Context independent

- Deep feedforward neural networks (FNN)

Previous work

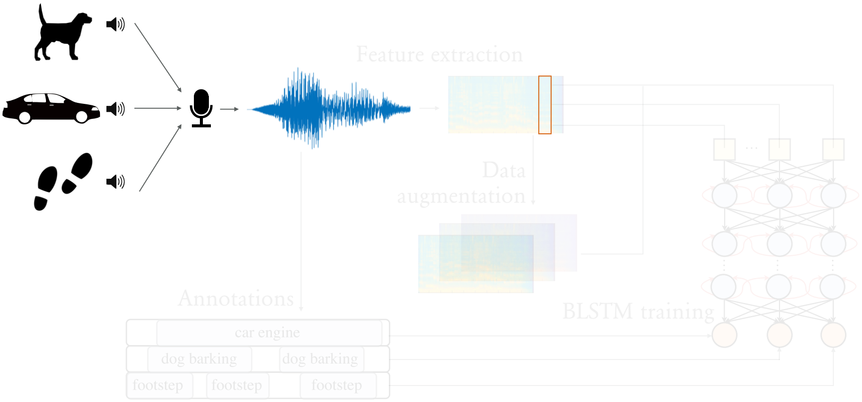
Context dependent

- GMM+HMM
- Nonnegative matrix factorisation (NMF)

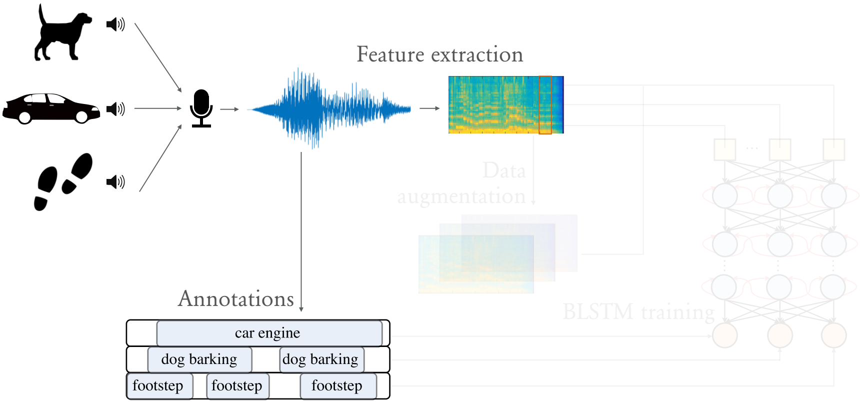
Context independent

- Deep feedforward neural networks (FNN)

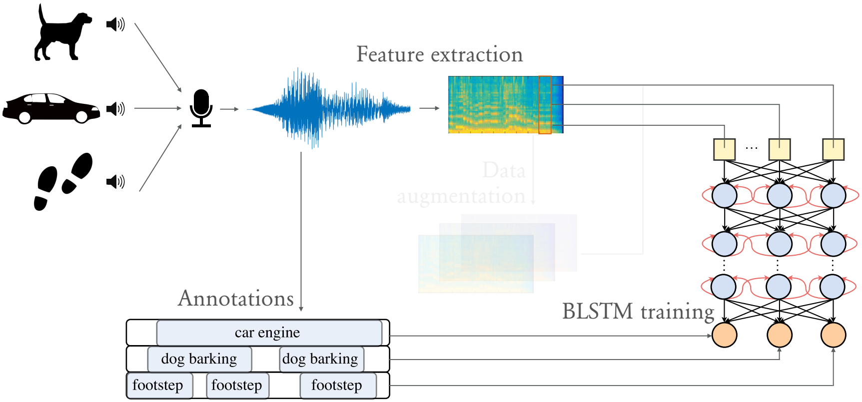
System overview



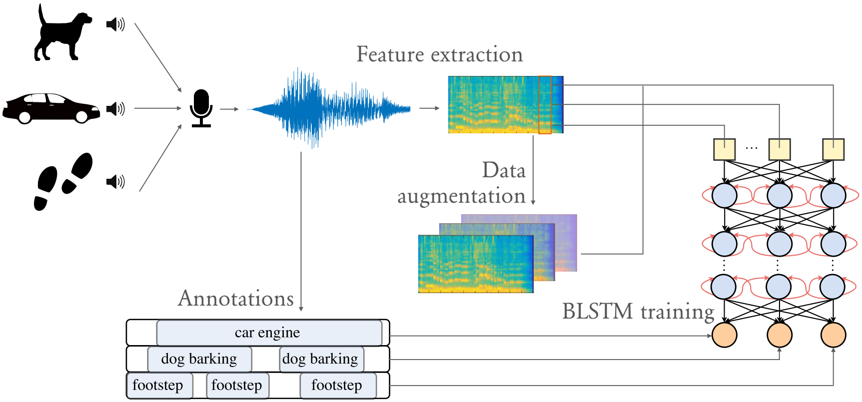
System overview



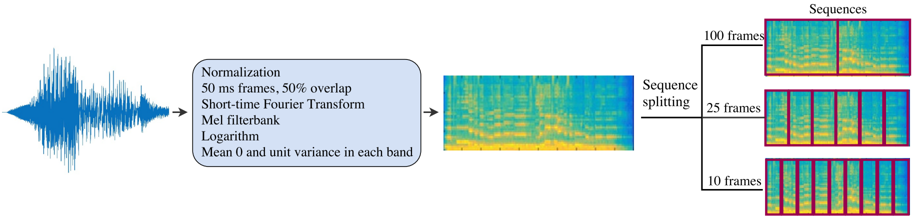
System overview



System overview



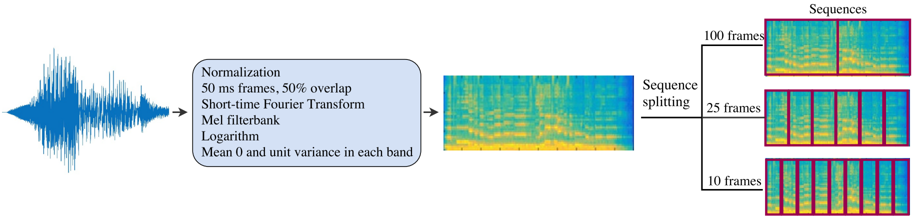
Feature extraction



Log mel energies, ZMUV, split in sequences at three different timescales (10, 25, 100 frames).

The data needs to be annotated: each class is marked as *active* (1) or *inactive* (0) in each frame.

Feature extraction



Log mel energies, ZMUV, split in sequences at three different timescales (10, 25, 100 frames).

The data needs to be annotated: each class is marked as *active* (1) or *inactive* (0) in each frame.

Recurrent neural network (RNN)

For an input sequence $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, compute a sequence of hidden activations $\{\mathbf{h}_1, \ldots, \mathbf{h}_T\}$ and output vectors $\{\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_T\}$ as

$$\mathbf{h}_t = \mathcal{F}(\mathbf{W}^{\mathrm{xh}}\mathbf{x}_t + \mathbf{W}^{\mathrm{hh}}\mathbf{h}_{t-1} + \mathbf{b}^{\mathrm{h}}) \tag{1}$$

$$\hat{\mathbf{y}}_t = \mathcal{G}(\mathbf{W}^{\mathrm{h}\hat{\mathrm{y}}}\mathbf{h}_t + \mathbf{b}^{\hat{\mathrm{y}}}) \tag{2}$$

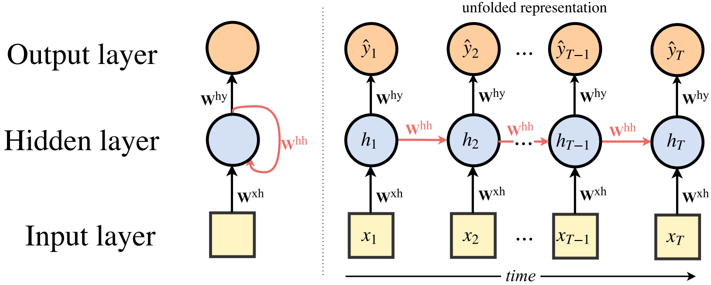


Figure: On the left, a recurrent neural network with 1 hidden layer and a single neuron. On the right, the same network unfolded in time over T steps.

Bidirectional RNN (BRNN) [1]

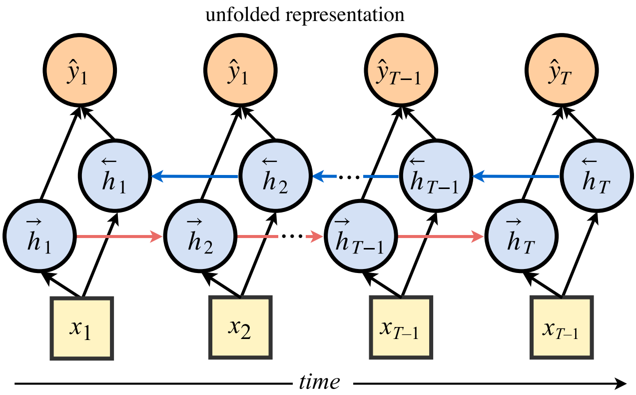
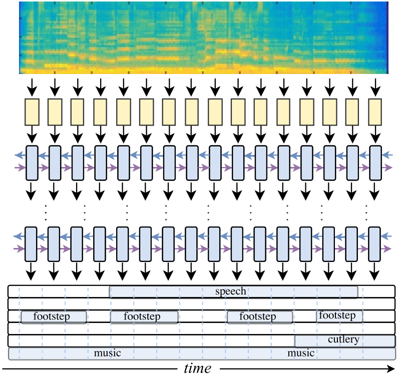


Figure: A bidirectional recurrent neural network with one hidden layer and two hidden neurons unfolded in time.

[1] Schuster et al., IEEE Trans. on Sgn. Processing (1997)

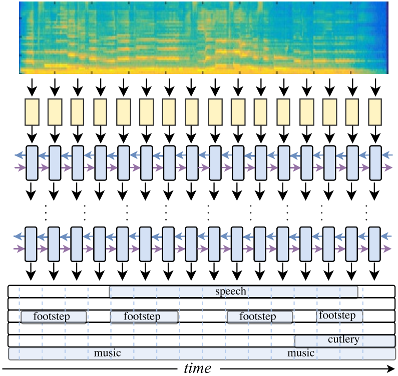
Model

- Bidirectional RNN with LSTM units (BLSTM)
- Multiple stacked recurrent hidden layers
- One output vector for each frame ("sequence to sequence")
- Output layer with sigmoids predicts posterior probabilities for each class of being active. Multilabel $\Longrightarrow$ no softmax
- At test time threshold predictions for binary activities



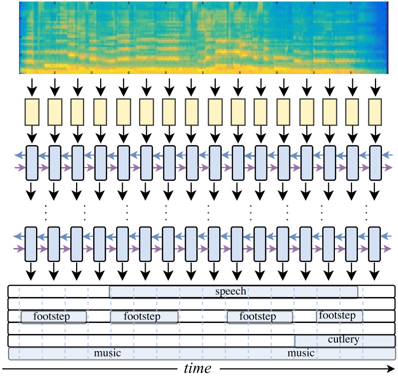
Model

- Bidirectional RNN with LSTM units (BLSTM)
- Multiple stacked recurrent hidden layers
- One output vector for each frame ("sequence to sequence")
- Output layer with sigmoids predicts posterior probabilities for each class of being active. Multilabel $\Longrightarrow$ no softmax
- At test time threshold predictions for binary activities



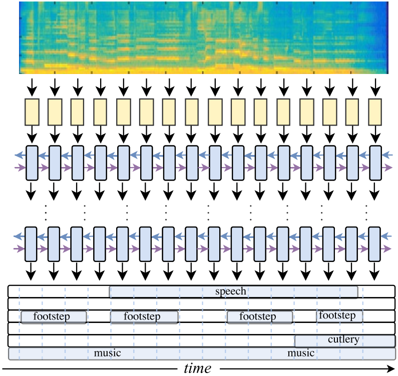
Model

- Bidirectional RNN with LSTM units (BLSTM)
- Multiple stacked recurrent hidden layers
- One output vector for each frame ("sequence to sequence")
- Output layer with sigmoids predicts posterior probabilities for each class of being active. Multilabel $\Longrightarrow$ no softmax
- At test time threshold predictions for binary activities



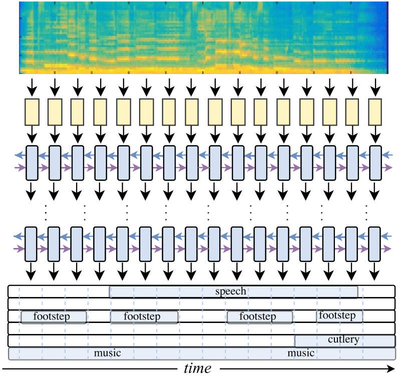
Model

- Bidirectional RNN with LSTM units (BLSTM)
- Multiple stacked recurrent hidden layers
- One output vector for each frame ("sequence to sequence")
- Output layer with sigmoids predicts posterior probabilities for each class of being active. Multilabel $\implies$ no softmax
- At test time threshold predictions for binary activities



Model

- Bidirectional RNN with LSTM units (BLSTM)
- Multiple stacked recurrent hidden layers
- One output vector for each frame ("sequence to sequence")
- Output layer with sigmoids predicts posterior probabilities for each class of being active. Multilabel $\Longrightarrow$ no softmax
- At test time threshold predictions for binary activities

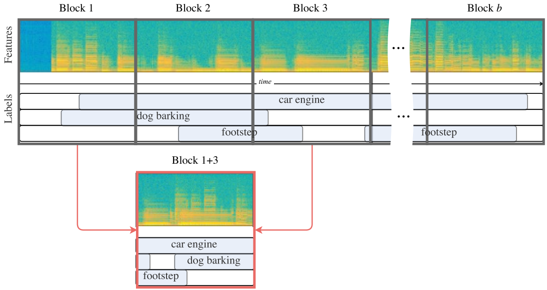


Data augmentation

Three techniques:

- Block mixing
- Time stretching
- Sub-frame time shifting

All performed directly in the time-frequency domain, on the extracted features.

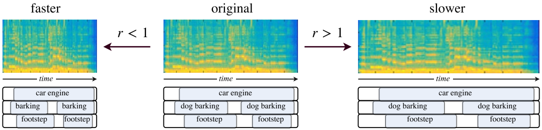


Data augmentation

Three techniques:

- Block mixing
- Time stretching
- Sub-frame time shifting

All performed directly in the time-frequency domain, on the extracted features.

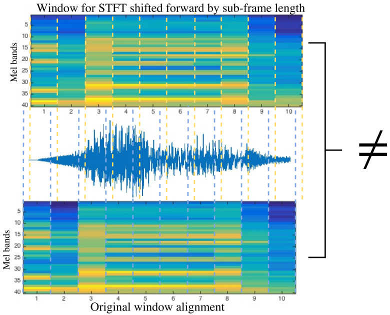


Data augmentation

Three techniques:

- Block mixing
- Time stretching
- Sub-frame time shifting

All performed directly in the time-frequency domain, on the extracted features.



Data augmentation

Three techniques:

- Block mixing
- Time stretching
- Sub-frame time shifting

All performed directly in the time-frequency domain, on the extracted features.

Data augmentation

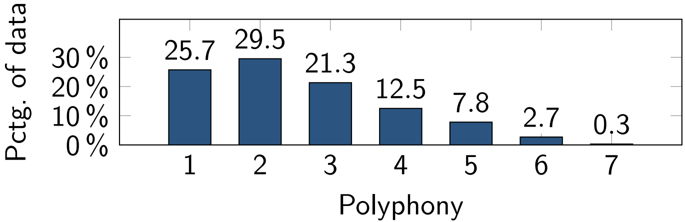
Three techniques:

- Block mixing
- Time stretching
- Sub-frame time shifting

All performed directly in the time-frequency domain, on the extracted features.

Dataset

CASA 61 classes from 10 contexts, real life recordings.
18 hours. 5 folds of training, validation and test.
Average polyphony 2.53

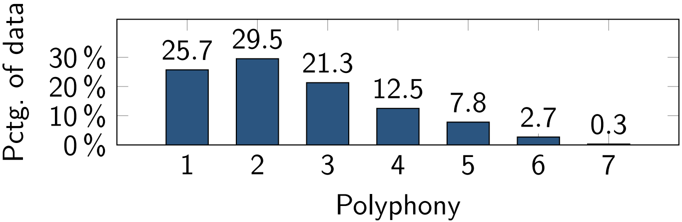


Augmentations:

- $\times$16 all combined (in the tables **+DA**)
  - Block mixing: 20 blocks per context, mixing 2 at the time $\times$9.5
  - Time stretching: stretching coeff {0.7, 0.85, 1.2, 1.5} $\times$4.25
  - Sub-frame time shifting: three times $\times$3

Dataset

CASA 61 classes from 10 contexts, real life recordings.
18 hours. 5 folds of training, validation and test.
Average polyphony 2.53



Augmentations:

- $\times 16$ all combined (in the tables **+DA**)
  - Block mixing: 20 blocks per context, mixing 2 at the time $\times 9.5$
  - Time stretching: stretching coeff $\{0.7, 0.85, 1.2, 1.5\}$ $\times 4.25$
  - Sub-frame time shifting: three times $\times 3$

Metrics

Overall metric is the average of the scores in each of the 10 contexts.

1. Framewise F1
2. 1-second F1

Neural nets used in the experiment

- 40 input neurons, each reading one band of the log mel energies
- 4 recurrent (BLSTM) hidden layers
- 200 LSTM blocks in each (100 forwards, 100 backwards)
- 850K parameters in total
- Optimizer: RMSprop
- Objective function: RMSE (cross entropy didn't work as well)
- Package: Currennt (CUDA/C++)
- 5 nets trained from random init for each fold, then pick the best on validation.
- At test time 100-frames sequences, threshold at 0.5
- Also tests using only LSTM (no bidirectional)
- No smoothing step required, RNN takes care of temporal continuity

Neural nets used in the experiment

- 40 input neurons, each reading one band of the log mel energies
- 4 recurrent (BLSTM) hidden layers
- 200 LSTM blocks in each (100 forwards, 100 backwards)
- 850K parameters in total
- Optimizer: RMSprop
- Objective function: RMSE (cross entropy didn't work as well)
- Package: Currennt (CUDA/C++)
- 5 nets trained from random init for each fold, then pick the best on validation.
- At test time 100-frames sequences, threshold at 0.5
- Also tests using only LSTM (no bidirectional)
- No smoothing step required, RNN takes care of temporal continuity

Neural nets used in the experiment

- 40 input neurons, each reading one band of the log mel energies
- 4 recurrent (BLSTM) hidden layers
- 200 LSTM blocks in each (100 forwards, 100 backwards)
- 850K parameters in total
- Optimizer: RMSprop
- Objective function: RMSE (cross entropy didn't work as well)
- Package: Currennt (CUDA/C++)
- 5 nets trained from random init for each fold, then pick the best on validation.
- At test time 100-frames sequences, threshold at 0.5
- Also tests using only LSTM (no bidirectional)
- No smoothing step required, RNN takes care of temporal continuity

Neural nets used in the experiment

- 40 input neurons, each reading one band of the log mel energies
- 4 recurrent (BLSTM) hidden layers
- 200 LSTM blocks in each (100 forwards, 100 backwards)
- 850K parameters in total
- Optimizer: RMSprop
- Objective function: RMSE (cross entropy didn't work as well)
- Package: Currennt (CUDA/C++)
- 5 nets trained from random init for each fold, then pick the best on validation.
- At test time 100-frames sequences, threshold at 0.5
- Also tests using only LSTM (no bidirectional)
- No smoothing step required, RNN takes care of temporal continuity

Neural nets used in the experiment

- 40 input neurons, each reading one band of the log mel energies
- 4 recurrent (BLSTM) hidden layers
- 200 LSTM blocks in each (100 forwards, 100 backwards)
- 850K parameters in total
- Optimizer: RMSprop
- Objective function: RMSE (cross entropy didn't work as well)
- Package: Currennt (CUDA/C++)
- 5 nets trained from random init for each fold, then pick the best on validation.
- At test time 100-frames sequences, threshold at 0.5
- Also tests using only LSTM (no bidirectional)
- No smoothing step required, RNN takes care of temporal continuity

Neural nets used in the experiment

- 40 input neurons, each reading one band of the log mel energies
- 4 recurrent (BLSTM) hidden layers
- 200 LSTM blocks in each (100 forwards, 100 backwards)
- 850K parameters in total
- Optimizer: RMSprop
- Objective function: RMSE (cross entropy didn't work as well)
- Package: Currennt (CUDA/C++)
- 5 nets trained from random init for each fold, then pick the best on validation.
- At test time 100-frames sequences, threshold at 0.5
- Also tests using only LSTM (no bidirectional)
- No smoothing step required, RNN takes care of temporal continuity

Neural nets used in the experiment

- 40 input neurons, each reading one band of the log mel energies
- 4 recurrent (BLSTM) hidden layers
- 200 LSTM blocks in each (100 forwards, 100 backwards)
- 850K parameters in total
- Optimizer: RMSprop
- Objective function: RMSE (cross entropy didn't work as well)
- Package: Currennt (CUDA/C++)
- 5 nets trained from random init for each fold, then pick the best on validation.
- At test time 100-frames sequences, threshold at 0.5
- Also tests using only LSTM (no bidirectional)
- No smoothing step required, RNN takes care of temporal continuity

Neural nets used in the experiment

- 40 input neurons, each reading one band of the log mel energies
- 4 recurrent (BLSTM) hidden layers
- 200 LSTM blocks in each (100 forwards, 100 backwards)
- 850K parameters in total
- Optimizer: RMSprop
- Objective function: RMSE (cross entropy didn't work as well)
- Package: Currennt (CUDA/C++)
- 5 nets trained from random init for each fold, then pick the best on validation.
- At test time 100-frames sequences, threshold at 0.5
- Also tests using only LSTM (no bidirectional)
- No smoothing step required, RNN takes care of temporal continuity

Neural nets used in the experiment

- 40 input neurons, each reading one band of the log mel energies
- 4 recurrent (BLSTM) hidden layers
- 200 LSTM blocks in each (100 forwards, 100 backwards)
- 850K parameters in total
- Optimizer: RMSprop
- Objective function: RMSE (cross entropy didn't work as well)
- Package: Currennt (CUDA/C++)
- 5 nets trained from random init for each fold, then pick the best on validation.
- At test time 100-frames sequences, threshold at 0.5
- Also tests using only LSTM (no bidirectional)
- No smoothing step required, RNN takes care of temporal continuity

Neural nets used in the experiment

- 40 input neurons, each reading one band of the log mel energies
- 4 recurrent (BLSTM) hidden layers
- 200 LSTM blocks in each (100 forwards, 100 backwards)
- 850K parameters in total
- Optimizer: RMSprop
- Objective function: RMSE (cross entropy didn't work as well)
- Package: Currennt (CUDA/C++)
- 5 nets trained from random init for each fold, then pick the best on validation.
- At test time 100-frames sequences, threshold at 0.5
- Also tests using only LSTM (no bidirectional)
- No smoothing step required, RNN takes care of temporal continuity

Neural nets used in the experiment

- 40 input neurons, each reading one band of the log mel energies
- 4 recurrent (BLSTM) hidden layers
- 200 LSTM blocks in each (100 forwards, 100 backwards)
- 850K parameters in total
- Optimizer: RMSprop
- Objective function: RMSE (cross entropy didn't work as well)
- Package: Currennt (CUDA/C++)
- 5 nets trained from random init for each fold, then pick the best on validation.
- At test time 100-frames sequences, threshold at 0.5
- Also tests using only LSTM (no bidirectional)
- No smoothing step required, RNN takes care of temporal continuity

Results

Comparing to the approach in [Cakir et al., 2015], which uses a FNN (MLP with maxout) with 1.6M parameters (double those of the RNN), where the outputs are smoothed using a median filter.

Table: Overall $F1$ scores, as average of individual contexts scores, for the FNN, the proposed LSTM and BLSTM, and BLSTM with data augmentation (+DA).

Method	$F1_{\text{AvgFram}}$	$F1_{\text{1-sec}}$
FNN [Cakir et al., 2015]	58.4%	63.0%
LSTM	62.5%	63.8%
BLSTM	64.0%	64.6%
BLSTM+DA	**64.7%**	**65.5%**

BLSTM+DA improves the performance over the FNN by relative 15.1% and 6.8% for $F1_{\text{AvgFram}}$ and $F1_{\text{1-sec}}$ respectively.

Results

Comparing to the approach in [Cakir et al., 2015], which uses a FNN (MLP with maxout) with 1.6M parameters (double those of the RNN), where the outputs are smoothed using a median filter.

Table: Overall $F1$ scores, as average of individual contexts scores, for the FNN, the proposed LSTM and BLSTM, and BLSTM with data augmentation (+DA).

Method	$F1_{\text{AvgFram}}$	$F1_{\text{1-sec}}$
FNN [Cakir et al., 2015]	58.4%	63.0%
LSTM	62.5%	63.8%
BLSTM	64.0%	64.6%
BLSTM+DA	**64.7%**	**65.5%**

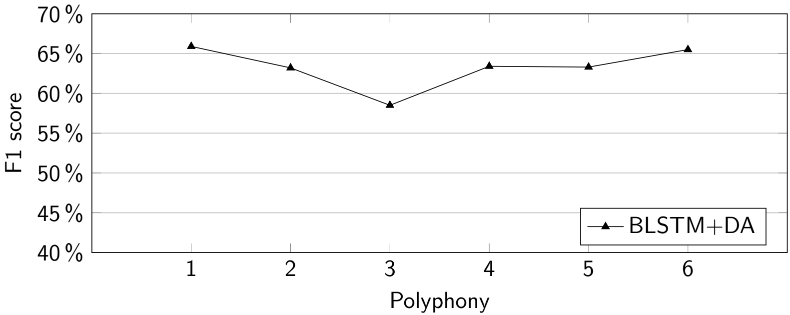
BLSTM+DA improves the performance over the FNN by relative 15.1% and 6.8% for $F1_{\text{AvgFram}}$ and $F1_{\text{1-sec}}$ respectively.

Results — individual contexts

	$F1_{\text{AvgFram}}$		
	FNN	**BLSTM**	**BLSTM+DA**
basketball	70.2%	77.4%	**78.5%**
beach	**49.7%**	46.6%	49.6%
bus	43.8%	45.1%	**49.4%**
car	53.2%	67.9%	**71.8%**
hallway	47.8%	**58.1%**	54.8%
office	77.4%	**79.9%**	74.4%
restaurant	69.8%	76.5%	**77.8%**
shop	51.5%	**61.2%**	61.1%
street	62.6%	**65.3%**	65.2%
stadium	58.2%	61.7%	**64.3%**
average	58.4%	64.0%	**64.7%**

Results — polyphony

Quite robust to polyphony increase



Demo

Demo time!

Discussion

1. RNNs improve over FNNs in polyphonic SED, and with half the parameters.
2. Overfitting, the main issue encountered $\implies$ much more data needed
3. Data augmentation helps slightly reducing overfitting.
4. Quite robust to high polyphony.

Discussion

1. RNNs improve over FNNs in polyphonic SED, and with half the parameters.
2. Overfitting, the main issue encountered $\Longrightarrow$ much more data needed
3. Data augmentation helps slightly reducing overfitting.
4. Quite robust to high polyphony.

Discussion

1. RNNs improve over FNNs in polyphonic SED, and with half the parameters.
2. Overfitting, the main issue encountered $\implies$ much more data needed
3. Data augmentation helps slightly reducing overfitting.
4. Quite robust to high polyphony.

Discussion

1. RNNs improve over FNNs in polyphonic SED, and with half the parameters.
2. Overfitting, the main issue encountered $\implies$ much more data needed
3. Data augmentation helps slightly reducing overfitting.
4. Quite robust to high polyphony.

The End

— Q&A —