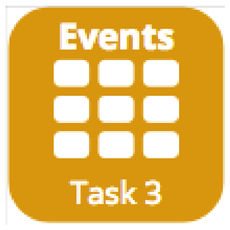
Audio event detection using multiple-input convolutional neural network

Il-Young Jeong[1,2], Subin Lee[1,2], Yoonchang Han[2], Kyogu Lee[1]

[1] Music and Audio Research Group, Seoul National University, Seoul, Korea [2] Cochlear.ai, Seoul, Korea

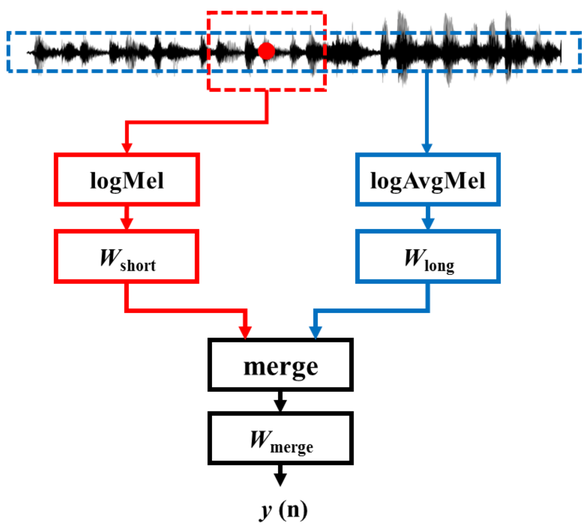
{iyjeong, sblee, ychan}@cochlear.ai, kglee@snu.ac.kr

Introduction



- This paper describes the model and training framework in our submission for **DCASE 2017 task 3: sound event detection in real life audio**. Extending the basic convolutional neural network architecture, we use both short- and long-term audio signal simultaneously as input data. In the training stage, we calculated validation errors more frequently than one epoch with adaptive thresholds. We also used class-wise early-stopping strategy to find the best model for each class. The proposed model showed meaningful improvements in cross-validation experiments compared to the baseline system.

Architecture



- **logMel**: log mel-spec. (1024 window and 729 shift)
- **logAvgMel**: log of frequency-wise average of mel-spec of full music track.
- ***W***: respective convnet. (consists of convolution and pooling)
- **merge**: combining the outputs of two layers (adding)
- **Optimizer**: Adam with 8 mini-batch size
- **Batch generation**:

  ±88, 573 stereo samples from the random offset (short-term)

  pre-computed logAvgMel (long-term)
- **Augmentation**: channel swapping

layer (short-term)	output size (filter×frame)	layer long-term	output size (filter×frame)
logMel	80×243	logAvgMel	40×1
conv	64×243	conv	64×1
pool	64×81	conv	64×1
conv	64×81	repeat	64×27
pool	64×27	↘	↙
	layer merged	output size (filter×frame)	
	add	64×27	
	conv	64×27	
	pool	64×9	
	conv	64×9	
	pool	64×3	
	fc	64×1	
	dropout	64×1	
	fc	6×1	

Learning strategy

- **Adaptive threshold**

  - 0.5 of detecting threshold may not be optimal.

  (Due to imbalance data distribution and different error function between training/validation)

  - It is empirically chosen for every class/validation, to minimize the error of validation data.

- **Class-wise early stopping**

  - Class-wise cost converged with different speed.

  - The optimal early-stopping point for one class may be too early (or too late) for the other classes.

  - Our solution: do early stopping individually.

  - In the evaluation (test) stage, classes are detected by using the respective model.

- **Frequent validation**

  - Model validation for every fixed number of mini-batch iteration.

  - Allows the proper early-stopping in training procedure whose validation error has fluctuation.

- **Cross-validation ensemble**

  - 4-fold CV models are combined for DCASE submission

  submission 1: majority vote (50% voting is considered as `active')

  submission 2: majority vote (50% voting is considered as `inactive')

  submission 3: majority vote without fold 1, that shows poor performance.

  submission 4: weighted vote based on those validation ERs.

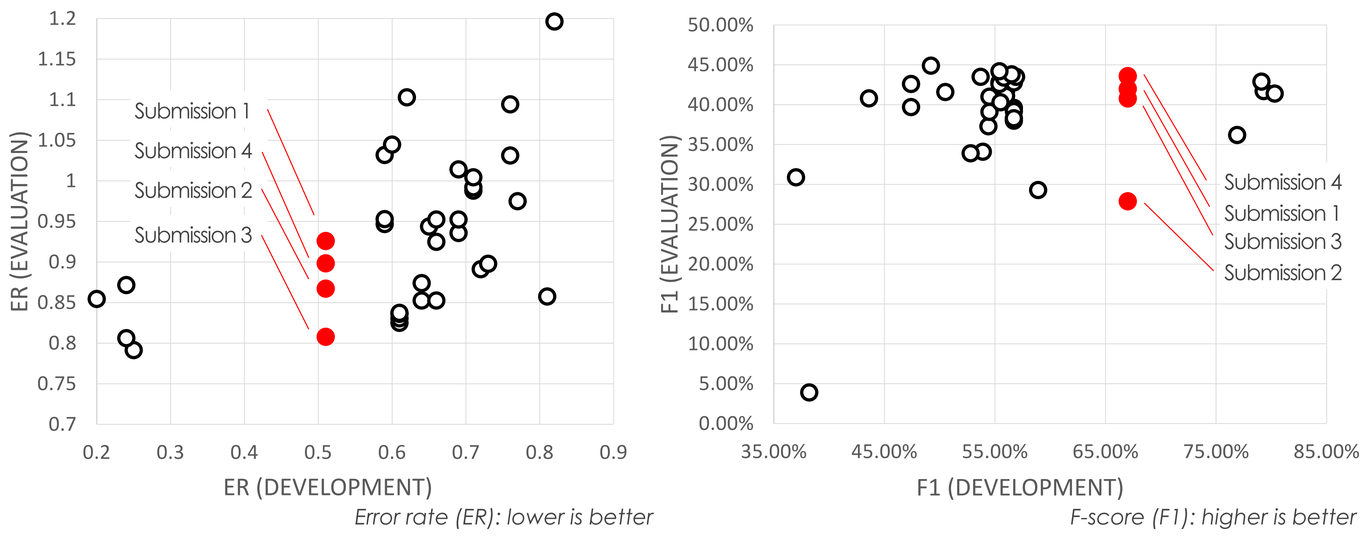
Results

- **Cross-validation results**

fold	1		2		3		4		average		baseline	
	ER	F	ER	F	ER	F	ER	F	ER	F	ER	F
brakes squeaking	1.00	0.0	1.00	0.0	0.97	11.8	0.93	32.4	0.98	11.1		
car	0.66	63.8	0.45	80.0	0.37	79.4	0.54	69.9	0.51	73.3		
children	1.00	0.0	1.00	0.0	0.99	2.7	0.25	85.7	0.81	22.1		
large vehicle	0.78	56.8	0.71	65.7	0.27	85.4	0.65	61.0	0.60	67.2		
people speaking	0.91	28.7	0.97	16.8	0.81	32.9	0.54	72.4	0.80	37.7		
people walking	0.92	19.3	0.25	86.0	0.46	76.8	0.53	68.6	0.54	62.7		
total	0.72	48.7	0.43	75.6	0.42	73.3	0.46	70.2	0.51	67.0	0.69	56.7

 # Cross-validation results may be affected by overfitting due to adaptive thresholding.

- **DCASE 2017 submission results**



Error rate (ER): lower is better *F-score (F1): higher is better*

Discussion and future work

- The following difficulties were encountered during the research process.
  - **imbalance data distribution**: we tried adaptive threshold, but it is unlikely to be a fundamental solution.
  - **multi-label classification**: It is not easy to use class-wise early stopping for many more classes (>100).
  - **annotation error**: we could not handle it in this study.
  - **data augmentation**, **noise-invariance**, **reducing computational cost for real-time detection**…

any ideas or tips?