DCASE 2017 SUBMISSION: MULTIPLE INSTANCE LEARNING FOR SOUND EVENT DETECTION

Justin Salamon[1,2]*, Brian McFee*[1,3]*, Peter Li*[1]*, Juan Pablo Bello*[1]*,*

[1] Music and Audio Research Laboratory
[2] Center for Urban Science and Progress
[3] Center for Data Science
New York University, New York, USA
{justin.salamon, brian.mcfee, phl232, jpbello}@nyu.edu

ABSTRACT

This extended abstract describes the design and implementation of a multiple instance learning model for sound event detection. The submitted systems use a convolutional-recurrent neural network (CRNN) architecture to learn strong (temporally localized) predictors from weakly labeled data using soft-max aggregation. Four variants of the proposed methods were submitted to DCASE 2017 [1], Task 4.

Index Terms— Deep learning, multiple instance learning, environmental sound

1. INTRODUCTION

Our approach to the weakly supervised sound event detection for smart cars (hereafter referred to as Task4) challenge is based on multiple instance learning [2]. Specifically, we developed convolutional-recurrent networks to predict strong (*i.e.*, time-varying) labels, which are aggregated to form weak (summary) predictions. The models are trained from weakly labeled data, and the learning signal is propagated through the temporal aggregation operator to inform the strong predictions.

Our model implementation is open source and publicly available to facilitate reproducibility.[1]

2. MODEL DESIGN

In this section, we describe the common architectural components of the submitted models.

2.1. Pre-processing

Each audio file was pre-processed with `librosa` 0.5 [3] and `pumpp` 0.3.1 [4]. Audio was converted to 44100 Hz mono, and converted to a log-power Mel spectrogram representation with a hop-length of 1024 samples, FFT window of 2048 samples, 128 Mel bins spanning 0 to 22050 Hz. For model development purposes, annotations were sampled at a matching frame-rate with `pumpp`.

2.2. CRNN architectures

The models under consideration all have a convolutional-recurrent neural network (CRNN) architecture, and use common processing blocks inspired by the audio component of the Look-Listen-Learn model [5], and Gated Recurrent Unit (GRU) networks [6].

Each model consists of a sequence of 4 *convolutional k-blocks*, defined as:

- k times:
  - n_i conv2D (3×3), ReLU activation, padded
  - batch normalization [7]
- max2D pooling (2×2) with stride of 2

where n_i denotes the number of filters in block i, and k denotes the number of convolutional layers in the block. In all models under consideration we set $n_i = 2^{3+i} = (16, 32, 64, 128)$.

Each k-block acts as a local feature extractor and dimensionality reduction module: each block reduces both the time and frequency resolution by a factor of 2. Because the input representation has $d = 128 = 2^7$, after 4 k-blocks, the frequency resolution is reduced to $2^3 = 8$ bands (and n_4 feature channels per band). A final conv2D layer of D filters with size $(1, 8)$ collapses the frequency dimension to produce a time-series of D-dimensional feature encodings.

The output of the final convolutional layer is then used as input to one or more bi-directional GRU layers. Here, we fixed the hidden-state dimension of each GRU to 128 (in each direction), resulting in a time-series of (128+128)-dimensional feature encodings $z(t)$.

Finally, the output of the (last) GRU layer is mapped through a sigmoid layer to produce a time-series of class predictions

$$\hat{y}(t) \in [0, 1]^C$$

corresponding to the C target labels for each frame t. This time-series is aggregated (see below) to produce a single class likelihood vector $\hat{Y}$ for the entire recording. The bias vector on the output layer is ℓ_2-regularized to limit sensitivity to skewed class distributions.

All models are trained using weak (track-level) annotations to optimize binary cross-entropy between Y and $\hat{Y}$, while the time-series predictions $\hat{y}(t)$ can be used at test time to temporally localize event detections.

2.3. Soft-max aggregation

In multiple instance learning, training data are labeled as *bags* of examples, where a bag is positive if any of its constituent examples are positive, and negative if none of its examples are positive.

[1] `https://github.com/justinsalamon/milsed`

Consequently, it is common to aggregate element-wise predictions (typically scores or likelihoods) by taking the maximum over all elements in the bag:

$$\hat{Y} = \max_t \hat{y}(t),$$

where $\hat{Y}$ is the bag-level prediction, and $\hat{y}(t)$ operate on the level of individual examples indexed by t. In the context of audio, a bag is typically a recording, and examples correspond to frames or short segments within the recording. In multi-label applications, the $\max$ is taken element-wise, independently for each output label.

While this design makes intuitive sense, it can be brittle in practice. During training, the $\max$ operator generally depends on a single element (frame), which in the context of deep learning methods, implies that the learning signal propagates only through that maximally activating frame, even if other frames are also highly activating for the target concept.

To circumvent this issue, we replace the $\max$ operator with the *soft max average* of predictions:

$$\tilde{Y}(X) = \sum_t \hat{y}(t) \left(\frac{\exp\left(\hat{y}(t)\right)}{\sum_u \exp\left(\hat{y}(u)\right)} \right).$$

This behaves similarly to the $\max$ operator, but allows gradients to flow through all frames in proportion to how highly they activate for each target concept.

In preliminary experiments, we compared max, soft-max average, and unweighted average pooling on the Task 4 validation data, and found soft-max averaging to outperform the others in all cases.

At test time, we can predict with either $\hat{Y}$ (weak-from-strong prediction) or $\tilde{Y}$ (weak-from-model prediction).

2.4. Ensembling

Given $M > 1$ pre-trained models, we also experimented with ensemble methods. The ensemble prediction $\hat{y}^*$ is constructed by taking the geometric mean of predicted class likelihood vectors from each element of the ensemble:

$$\hat{y}^*(t) := \exp\left(\frac{1}{M} \sum_{i=1}^{M} \log \hat{y}_i(t) \right)$$

where $\hat{y}_i$ is the prediction from the ith model.

For weak-from-strong prediction, the ensemble prediction $\hat{Y}^*$ is taken as the max over the ensembled strong predictions:

$$\hat{Y}^* := \max_t \hat{y}^*(t)$$

For weak-from-model prediction, the ensemble prediction $\tilde{Y}^*$ is the goemetric mean of the individual weak predictions $\tilde{Y}_i$:

$$\tilde{Y}^* := \exp\left(\frac{1}{M} \sum_{i=1}^{M} \log \tilde{Y}_i \right).$$

3. IMPLEMENTATION

All of our models were developed in `Keras` 2.0 [8] and `TensorFlow` 1.1 [9]. Training was facilitated with `Pescador` 1.0 [10].

3.1. Data augmentation

To improve the robustness of the models, we include training with data augmentation using the `muda` library 0.1.4 [11, 12]. Specifically, we experimented with pitch transposition (up to ± 5 semitones) and dynamic range compression (DRC) augmentation using preset modes *radio, film standard, music standard, speech*.

3.2. Training setup

The weakly labeled training set was randomly split into 75%/25% training and validation. Models were trained using the Adam optimizer [13] with mini-batches of 32 examples per batch. Each training epoch consists of 512 batches, and validation was performed by randomly sampling 1024 batches. The learning rate was decreased automatically if the validation score has not decreased in 10 epochs, and training was terminated early if it did not decrease in 30 epochs. Training was limited to 150 epochs total.

3.3. Validation and model selection

We experimented with a variety of architectures, varying the block size $k \in \{2, 3, 5\}$ and the number of GRU layers $\{1, 2, 3\}$, as well as the aggregation method ($\max$, mean, or soft-max), and form of data augmentation (pitch and/or DRC). Models were evaluated on the validation set for weak $F1$-macro and -micro average across categories, and on the small, strongly labeled test set for strong predictions ($F1$-scores and error rate).

We also experimented with including CBHG layers [14] between the convolutional and recurrent layers, but this was not found to provide significant improvement over the simpler CRNN models.

For generating model ensembles, we evaluated all 120 non-trivial subsets of a collection of 7 candidate configurations.

4. SUBMITTED SYSTEMS

In our final submissions, we included four configurations, listed in Table 2. Two submitted models (systems 1 and 2) are individual predictors, whose architectures are described in Table 1. The remaining two models (3 and 4) are ensembles, whose components are also described in Table 1.

The final selected configurations were chosen to achieve high performance ($F1$-score and error rate), and test the influence of specific design choices:

- 1 vs. 2: soft-max or mean aggregation?
- 1 vs. 3: ensemble or single-model?
- 3 vs. 4: weak-from-strong or weak-from-model?

4.1. Results

The performance of the systems on the development test-set for weak (subtask A) and strong (subtask B) prediction are listed in Table 3. Weak labeling metrics are computed with `scikit-learn` [15] using micro-averaging (instance based). Strong labeling metrics are computed with `sed_eval` [16] using segment-based evaluation with a segment duration of 1 second.

Table 1: Individual model configurations. Block size corresponds to the number of convolutional layers per processing block.

Model	Block size	# GRU	# Parameters	Aggregation	Augmentation
A	$k = 2$	1	858K	soft-max	pitch
B	$k = 2$	3	1.45M	soft-max	pitch
C	$k = 2$	3	1.45M	mean	pitch
D	$k = 3$	3	1.65M	soft-max	DRC + pitch

Table 2: Systems submitted for Task 4. See Table 1 for details of each model.

Name	Model(s)	Weak prediction rule
System 1	B	from strong
System 2	C	from strong
System 3	A,B,D	from strong
System 4	A,B,D	from model

5. ACKNOWLEDGMENT

We would like to thank the administrators of the DCASE2017 challenge for all of their hard work in running the system, and dutifully answering all of our annoyingly detailed questions.

6. REFERENCES

[1] http://www.cs.tut.fi/sgn/arg/dcase2017/.

[2] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial intelligence*, vol. 89, no. 1, pp. 31–71, 1997.

[3] B. McFee, M. McVicar, O. Nieto, S. Balke, C. Thome, D. Liang, E. Battenberg, J. Moore, R. Bittner, R. Yamamoto, D. Ellis, F.-R. Stoter, D. Repetto, S. Waloschek, C. Carr, S. Kranzler, K. Choi, P. Viktorin, J. F. Santos, A. Holovaty, W. Pimenta, and H. Lee, "librosa 0.5.0," Feb. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.293021

[4] B. McFee, "pumpp: a practically universal music preprocessor," July 2017. [Online]. Available: https://doi.org/10.5281/zenodo.839941

[5] R. Arandjelovic and A. Zisserman, "Look, listen and learn," *CoRR*, vol. abs/1705.08168, 2017. [Online]. Available: http://arxiv.org/abs/1705.08168

[6] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[7] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[8] F. Chollet *et al.*, "Keras," 2015.

[9] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[10] B. McFee, C. Jacoby, and E. Humphrey, "pescador," Mar. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.400700

[11] B. McFee and M. Cartwright, "bmcfee/muda: 0.1.4," July 2017. [Online]. Available: https://doi.org/10.5281/zenodo.830018

[12] B. McFee, E. Humphrey, and J. Bello, "A software framework for musical data augmentation," in *16th International Society for Music Information Retrieval Conference*, ser. ISMIR, 2015.

[13] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[14] J. Lee, K. Cho, and T. Hofmann, "Fully character-level neural machine translation without explicit segmentation," *arXiv preprint arXiv:1610.03017*, 2016.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1953048.2078195

[16] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016. [Online]. Available: http://www.mdpi.com/2076-3417/6/6/162

Table 3: Validation- and test-set results for all four submitted model configurations, under both weak (W) and strong (S) evaluation conditions. F1 scores are micro-averaged. ER is the *error-rate* metric for strong prediction: lower scores are better.

System	(W) F1	(W) Precision	(W) Recall	(S) F1	(S) Precision	(S) Recall	ER
1	0.459	0.447	0.470	0.404	0.423	0.387	0.843
2	0.440	0.399	0.490	0.393	0.405	0.383	0.861
3	0.455	0.537	0.394	0.410	0.533	0.334	0.761
4	0.380	0.630	0.272	0.410	0.533	0.334	0.761