

MULTI-VIEW AUDIO AND MUSIC CLASSIFICATION

Huy Phan^{*1}, Huy Le Nguyen², Oliver Y. Chén³, Lam Pham⁴, Philipp Koch⁵,
Ian McLoughlin⁶, Alfred Mertins⁵

¹Queen Mary University of London, UK, ²HCMC University of Technology, Vietnam

³University of Oxford, UK, ⁴Austrian Institute of Technology Vienna, Austria

⁵University of Lübeck, Germany, ⁶Singapore Institute of Technology, Singapore

*Correspondence email: h.phan@qmul.ac.uk

ABSTRACT

We propose in this work a multi-view learning approach for audio and music classification. Considering four typical low-level representations (i.e. different views) commonly used for audio and music recognition tasks, the proposed multi-view network consists of four subnetworks, each handling one input types. The learned embedding in the subnetworks are then concatenated to form the multi-view embedding for classification similar to a simple concatenation network. However, apart from the joint classification branch, the network also maintains four classification branches on the single-view embedding of the subnetworks. A novel method is then proposed to keep track of the learning behavior on the classification branches and adapt their weights to proportionally blend their gradients for network training. The weights are adapted in such a way that learning on a branch that is generalizing well will be encouraged whereas learning on a branch that is overfitting will be slowed down. Experiments on three different audio and music classification tasks show that the proposed multi-view network not only outperforms the single-view baselines but also is superior to the multi-view baselines based on concatenation and late fusion.

Index Terms— multi-view learning, deep learning, audio classification, music classification, gradient blending

1. INTRODUCTION

Good embeddings are crucial for machine learning tasks [1, 2, 3]. For audio and music classification, in particular, such an embedding can be learned from a variety of low-level features which have been developed alongside the development of the research field, such as Mel-scaled spectrogram [4, 5, 6, 7], Gammatone spectrogram [8, 9, 2], Constant-Q transform (CQT) spectrogram [10, 11, 12], and even raw waveform [13, 14]. Oftentimes, recognition results obtained from embeddings learned from different low-level inputs vary in the sense that one embedding is good for some target classes while another is good for some other target classes. This implies that the embeddings are complement and the low-level inputs can be reasonably considered as different views of the target data. Intuitively, owing to their complementarity, jointly learning from these views should leverage their individual strength and gives rise to performance gain on a task at hand [15, 16]. However, it is not always the case in practice as a naive fusion scheme, e.g. concatenation [2, 17, 18], may result in performance degradation rather than improvement, i.e. the multi-view performance could be worse than that of the best single view [17, 18]. The reason is that different single-view subnetworks learn at different rates and converge/overfit

at different times during the training course. As a result, fusing out-of-sync single-view subnetworks via concatenation results in a sub-optimal multi-view model. Late fusion is another common approach for fusing information from multiple views; however, separate training single-view networks is unable to take into account interaction between the views.

Inspired by prior work in [17, 18], we propose a novel multi-view learning method based on deep learning for audio and music classification that overcomes the aforementioned issues. In the proposed approach, a multi-view network is designed so that we are able to gain access to the convergence/overfitting behavior of the constituent single-view subnetworks. This then allows us to individualize their learning during the training process. In intuitive, learning on subnetworks that are generalizing well is encouraged whereas learning on subnetworks that are overfitting is slowed down. This is accomplished by assigning different weights to the subnetworks' losses prior to blending their gradients [17, 18]. The weights are adaptively adjusted according to the subnetworks' learning behavior. By doing this, we are able to regulate the contribution of each view into the multi-view embedding rather than even their contribution as in the case of simple concatenation. Our experiments on three different audio and music classification tasks (environmental sound classification, audio scene classification, and music genre classification) show that the multi-view embedding learned via the proposed method consistently results in better performance than that obtained by all the single-view baselines and the multi-view baselines based on concatenation and late fusion.

2. LEARNING MULTI-VIEW AUDIO/MUSIC EMBEDDING

2.1. Network architecture

We adopt four low-level features, including Mel-scale spectrogram, Gammatone spectrogram, CQT spectrogram, and raw waveform, which are most widely used for audio and music analysis under deep learning paradigms. They are considered as different views of the underlying data distribution of a audio/music classification task at hand. The proposed network for learning multi-view embedding is illustrated in Fig. 1. It consists of four subnetworks, each of which is to process one of the low-level inputs. The multi-view embedding is formed by concatenating the embeddings learned by the view-specific subnetworks. However, apart from the concatenation branch, the view-specific CRNNs also maintain their own classification branches which serve as a gateway to access their learning behavior. The subnetworks are realized by convolutional recurrent neural networks (CRNNs) that are described below.

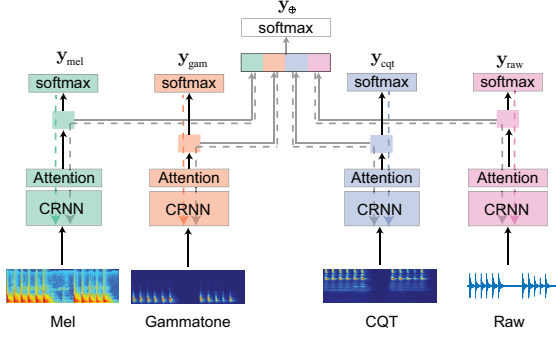


Fig. 1. Illustration of the network for learning multi-view embedding. The dash lines represent the gradient backpropagation flows.

2D CRNNs: The 2D inputs (i.e. Mel-scale, Gammatone, and CQT spectrogram) have a general size of $T \times F$ where T is the number of time frames and F frequency bands. The CRNNs corresponding to the 2D inputs share a similar network architecture whose configuration is shown in Table 1. The architecture features six convolutional layers, each associated with Rectified Linear Unit (ReLU) activation [19], batch normalization [20], and a max pooling layer. The max pooling layers have a common kernel size of 2×1 and stride 1×1 to reduce size of spectral dimension by half while maintaining the temporal dimension of the input. On top of the convolutional layers, a bidirectional recurrent neural network (biRNN) layer is employed for sequential modelling on the time dimension. It is realized by Gated Recurrent Units (GRUs) [21]. The sequence of recurrent outputs is then reduced to a feature vector (i.e. view-specific embedding) via spatio-temporal attention pooling suggested in [8]. For classification purpose, the 2D CRNNs make use of two fully-connected layers with ReLU activation, followed by a final output layer with softmax. A dropout rate of 0.1 is applied to the convolutional layers, the recurrent layers, and the fully-connected layers.

1D CRNN: The 1D CRNN’s configuration is shown in Table 2. Out of eight convolutional layers, the first two (*conv01* and *conv02*) coupled with the max pooling layer (*pool02*) are tailored to transform the raw input into a 2D representation as in [14, 13]. The rest of the network can then be parametrized similar to the 2D CRNNs described above, except for the *conv1* and *pool1* for which larger (temporal) kernel sizes are set to efficiently deal with the larger (temporal) input as well as to shorten its temporal dimension.

Beside the single-view classification branches on the CRNNs, classification on the multi-view embedding is carried out via two fully-connected layers with 4096 hidden units and ReLU activation, followed by a final output layer with softmax. Let \mathbf{y} denote the one-hot encoding ground-truth and $\hat{\mathbf{y}}^{(k)}$ denote the output of the classification branch $k \in \{mel, gam, cqt, raw, \oplus\}$. We use \oplus to denote the multi-view classification branch. The cross-entropy loss induced by the branch k on a set of M samples reads:

$$\mathcal{L}^{(k)} = -\frac{1}{M} \sum_{m=1}^M \mathbf{y}_m \log(\hat{\mathbf{y}}_m^{(k)}). \quad (1)$$

The total loss used for training at the training step n is computed by:

$$\mathcal{L}(n) = \sum_k w^{(k)}(n) \mathcal{L}^{(k)}(n), \quad (2)$$

where $w^{(k)}(n)$ denotes the weight of the classification branch k at the training time n . $w^{(k)}(n)$ is adapted during the training process according to the learning behavior of the branch k .

Table 1. Configuration of the 2D CRNNs. The output shape is of the format (*time, frequency, channel*). Here, the number of kernels $F_l = (32, 64, 128, 128, 256, 512)$ for six convolutional layers indexed by $l = (1, 2, 3, 4, 5, 6)$.

Layer	Filter size	Stride	#filters	Padding	Output shape
Input					$(T, 64, 1)$
conv- l	$(3, 3)$	$(1, 1)$	F_l	SAME	$(T, \frac{64}{2^{l-1}}, F_l)$
pool- l	$(1, 2)$	$(1, 1)$		VALID	$(T, \frac{64}{2^l}, F_l)$
reshape					$(T, 512)$
biRNN			$2 \cdot 256$		$(T, 512)$
attention			64		$(512,)$
fc1			1024		$(1024,)$
fc2			1024		$(1024,)$
fc3			#classes		$(\#classes,)$

Table 2. Configuration of the 1D CRNN. The output shape is of the format (*time, frequency, channel*). Here, the number of kernels $F_l = (64, 128, 128, 256, 512)$ for five convolutional layers indexed by $l = (2, 3, 4, 5, 6)$.

Layer	Filter size	Stride	#filters	Padding	Output shape
Input					$(66, 650, 1, 1)$
conv01	$(64, 1)$	$(2, 1)$	32	VALID	$(16, 640, 1, 64)$
conv02	$(16, 1)$	$(2, 1)$	64	VALID	$(260, 1, 64)$
pool02	$(64, 1)$	$(64, 1)$		VALID	$(260, 64, 1)$
reshape					$(260, 64, 1)$
conv1	$(5, 3)$	$(1, 1)$	32	SAME	$(260, 64, 32)$
pool1	$(4, 2)$	$(4, 2)$		VALID	$(65, 32, 32)$
conv- l	$(3, 3)$	$(1, 1)$	F_l	SAME	$(65, \frac{64}{2^{l-1}}, F_l)$
pool- l	$(1, 2)$	$(1, 1)$		VALID	$(65, \frac{64}{2^l}, F_l)$
reshape					$(65, 512)$
biRNN			$2 \cdot 256$		$(65, 512)$
attention			64		$(512,)$
fc1			1024		$(1024,)$
fc2			1024		$(1024,)$
fc3			#classes		$(\#classes,)$

2.2. Adaptive gradient blending

Similar to [17, 18], learning behavior on the branch k can be assessed via the generalization measure $G^{(k)}$ and the overfitting measure $O^{(k)}$. In intuition, $G^{(k)}$ represents the information about the target distribution gained via training and $O^{(k)}$ represents the gap between information gain on the training set and the target distribution. $G^{(k)}$ and $O^{(k)}$ at the training step n are approximated as:

$$G^{(k)}(n) \approx L_{\diamond}^{*(k)} - L_{\diamond}^{(k)}(n), \quad (3)$$

$$O^{(k)}(n) \approx (L_{tr}^{*(k)} - L_{tr}^{(k)}(n)) - (L_{\diamond}^{*(k)} - L_{\diamond}^{(k)}(n)). \quad (4)$$

In (3) and (4), $L_{tr}^{(k)}(n)$ and $L_{\diamond}^{(k)}(n)$ denote the loss on a training set and the loss on a test set (i.e. the true loss) at the training step n , respectively. $L_{tr}^{*(k)}$ and $L_{\diamond}^{*(k)}$ denote the training and true loss references, respectively. Since the true loss is unknown, we approximate it by the loss on a validation set. The weight $w^{(k)}$ for the branch k is then computed as the ratio of generalization and overfitting measure:

$$w^{(k)}(n) = \frac{1}{Z} \frac{G^{(k)}(n)}{O^{(k)2}(n)}, \quad (5)$$

where Z is a normalization factor. A network branch which is generalizing (i.e., large G^k and small O^k) will have a larger weight to encourage its learning. In contrast, a network branch which is overfitting (i.e., small G^k and large O^k) will have a smaller weight to discourage its learning. A square for O^k in (5) is to avoid the situation when an underfitting network branch still scores very well on the generalization-over-overfitting ratio and receives a large weight.

Algorithm 1 Computation of an adaptive weight

```
1: procedure ADAPTIVEWEIGHT( $L_{tr}, L_{\diamond}, L_{tr}^*, L_{\diamond}^*, W$ )
2:   Input:  $L_{tr}[1 \dots n]$ : list of training loss values
3:    $L_{\diamond}[1 \dots n]$ : list of true loss values
4:    $L_{tr}^*$ : current best training loss value
5:    $L_{\diamond}^*$ : current best true loss value
6:    $W$ : smoothing window size
7:   Output:  $w(n)$ : weight at the training time  $n$ 
8:    $\bar{L}_{tr}(n) = \text{mean}(L_{tr}[(n-W) \dots n])$   $\triangleright$  Smoothed training loss
9:    $\bar{L}_{\diamond}(n) = \text{mean}(L_{\diamond}[(n-W) \dots n])$   $\triangleright$  Smoothed true loss
10:   $G(n) = L_{\diamond}^* - \bar{L}_{\diamond}(n)$   $\triangleright$  Eq. (3)
11:   $O(n) = (L_{tr}^* - L_{tr}(n)) - (\bar{L}_{tr}^* - \bar{L}_{tr}(n))$   $\triangleright$  Eq. (4)
12:   $w(n) = \frac{1}{Z} \frac{G(n)}{O^2(n)}$   $\triangleright$  Eq. (5)
13:  if  $\bar{L}_{tr} < L_{tr}^*$  then  $L_{tr}^* = \bar{L}_{tr}$   $\triangleright$  Update best training loss
14:  if  $\bar{L}_{\diamond} < L_{\diamond}^*$  then  $L_{\diamond}^* = \bar{L}_{\diamond}$   $\triangleright$  Update best true loss
```

Equations (3) and (4) suggest that the accuracy of the approximations for $G^{(k)}$ and $O^{(k)}$ depends on the references $L_{tr}^{*(k)}$ and $L_{\diamond}^{*(k)}$. In [17, 18], the losses $L_{tr}^{(k)}(0)$ and $L_{\diamond}^{(k)}(0)$ at time $n = 0$ (i.e. right after the network initialized with random weights) were used for this purpose. However, we empirically found that these fixed references resulted in unsatisfactory performance. We conjecture that it is most likely due to the bias to a specific random initialization of the network (i.e. different random initializations will lead to various approximation accuracy). To overcome this, we propose to use the best losses up the current time n for references. Furthermore, these references are also adapted during the training course. Furthermore, in audio classification tasks, even though the overall trend of the loss curves are smooth, they are noisy in short term, we therefore smooth the losses with a history window of size W before updating the loss references to avoid being stuck in local minima. The procedure for computing the weight $w^{(k)}$ is devised in Algorithm 1.

2.3. Self-ensemble

Since the multi-view network has multiple outputs (i.e. the single-view classification outputs and the multi-view classification output) which can be aggregated to produce a self-ensemble of decisions:

$$P(y = c) = \frac{1}{5} \sum_k \left(w_{\diamond}^{(k)} P^{(k)}(y = c) \right), \quad (6)$$

where $P^{(k)}(y = c)$ denotes the probability that the classification branch k predicts the category $c \in \{1, \dots, C\}$ out of C categories. We use $w_{\diamond}^{(k)}$ to denote the weight of the classification branch k found with the final model. The final output label is then determined as:

$$\hat{y} = \underset{c}{\operatorname{argmax}} P(y = c). \quad (7)$$

3. EXPERIMENTS

3.1. Experimental setup

3.1.1. Datasets

We employed three databases to conduct experiments on three audio and music classification tasks: environmental sound classification, audio scene classification, and music genre classification.

ESC-50 [6]: This dataset consists of 2,000 monaural samples in total which are evenly distributed among 50 environmental sound categories. Each sample has a length of roughly 5 seconds sampled at 44.1 kHz. The dataset was divided into 5 folds and we adhered to [6] to conduct 5-fold cross validation.

DCASE2016 Task 1 [22]: This dataset was used in the audio scene classification (Task 1) of the DCASE 2016 challenge [22]. It consists of 1,560 binaural samples evenly distributed among 16 audio scene categories. The data was recorded with a sampling frequency of 44.1 kHz and each sample has a length of 30 seconds. We used the development set for training and the evaluation set for testing in the experiments. Note that, for simplicity, binaural audio was reduced to monaural before experimentation.

GTZAN [23]: This dataset has been widely used for evaluation of music genre classification. It consists of 10 genres with 100 audio files each, all having a length of 30 seconds and sampling frequency of 22,050 Hz. We conducted 10-fold cross validation following [23].

3.1.2. Feature extraction

To extract the 2D low-level features, a raw audio signal was transformed into a log Mel-scale spectrogram using $F = 64$ Mel-scale filters in the frequency range up to Nyquist rate. Similarly, log Gammatone spectrogram was extracted using $F = 64$ Gammatone filters. A window size of 40ms and 50% overlap were commonly used. Log CQT spectrogram [24] was extracted using Librosa [25] with $F = 64$ frequency bins, 12 bins per octave, and a hop length of 512 (for 22,050 Hz sampling rate) or 1024 (for 44.1 kHz sampling rate).

Note that with this setting, the time dimension of the CQT spectrogram is smaller than that of the Mel-scale and Gammatone ones. A 30-second snippet at 44.1 kHz sampling rate results in a Mel-scale and a Gammatone spectrogram of size 1499×64 while the resulted CQT spectrogram is of size 1292×64 .

3.1.3. Parameters

The network was trained using Adam optimizer [26] for $E = 3000$ epochs (ESC-50) and $E = 1500$ epochs (DCASE 2016 and GTZAN) with a minibatch size of 64. We used the Mel-scaled and Gammatone inputs of length $T = 75$ frames, the CQT input of length of $T = 65$ frames, and the raw waveform input of length 66,650 samples (with 44.1 kHz sampling rate) or 33,330 samples (with 22,050 Hz sampling rate). It should be noted that when the raw input has length of 33,330, the *pool02* layer in the 1D CRNN (cf. Table 2) had its kernel size and stride reduced by half.

The learning rate was initially set to 2×10^{-4} and was exponentially reduced with a rate of 0.8 after $0.1E$, $0.2E$, and $0.3E$ epochs. In addition, the first 10 epochs were used as a warm-up period in which the network was trained with a small learning rate of 2×10^{-5} . For model selection and for approximating the true loss in (3) and (4), a validation set was randomly drawn and left out. More specifically, samples from two audio sources per category in case of ECS-50, samples from 10% of audio sources per category in case of DCASE2016, and 10% of samples in case of GTZAN were used for this purpose. During training, the network that resulted in best validation accuracy was retained. Note that, in order to compute the training loss in (4), evaluating the network on the entire training set would be computationally expensive. Instead, we sampled and fixed a small subset of training examples (roughly the same size as the validation set) for approximation.

During testing, for an audio file of length S seconds, S data samples were evenly sampled and presented to the trained network for classification. The global classification decision was obtained by aggregating the segment-wise decisions via averaging.

3.1.4. Baselines

To assess the efficacy of the proposed multi-view method, we constructed four single-view baselines and two multi-view baselines for

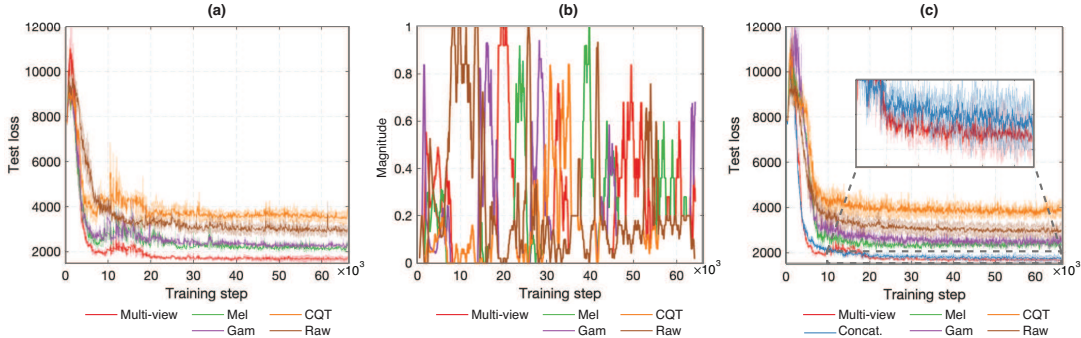


Fig. 2. ESC-50: (a) The test loss curves (averaged over 5 cross-validation folds) of the classification branches of the multi-view network; (b) The weights assigned to the classification branches of the multi-view network during training (only the first cross-validation fold is shown); (c) The test loss curves (averaged over 5 cross-validation folds) of the multi-view network and the baselines during training.

comparison. The four single-view baselines were the CRNN subnetworks in Fig. 1 that were trained independently on the individual low-level inputs. The first multi-view baseline had a similar architecture to the proposed multi-view network (cf. Fig. 1) but relied on simple concatenation fusion. The second multi-view baseline was a late-fusion system that combined the independent single-view baselines by taking average of their classification probabilities.

3.2. Experimental results

The classification accuracies obtained by the proposed multi-view method and the baselines over the experimental databases are shown in Table 3. On the one hand, among the single-view baselines, the ones using the Mel-scale and Gammatone spectrogram inputs results in better performance than those using the CQT spectrogram and raw inputs. This result is consistent with the finding in [11] and also reflects the fact that the former two are more popular than the latter two in various audio/music recognition tasks. Although combining multiple views via the simple concatenation and late fusion leads to performance gains in all the experimental databases, it is disputable whether late fusion works better than concatenation since the former outperforms the latter on ESC-50 and GTZAN whereas the opposite result was seen on DCASE2016.

On the other hand, the proposed multi-view network consistently outperforms not only the single-view baselines but also the multi-view baselines over all three tasks. More specifically, our network achieves an accuracy gain of 0.85%, 1.28%, and 1.1% absolute on ESC-50, DCASE2016, and GTZAN over the best baseline (i.e. late fusion, concatenation, and late fusion), respectively. The superiority of the proposed method is also reflected by its lower test loss as shown in Fig. 2(c). The gain via self-ensembling is even better, achieving 1.55%, 1.53%, and 1.4% absolute, respectively.

These results suggest that the proposed multi-view learning method is more efficient than the popular concatenation and late fusion methods. It can be explained that the proposed method offers a mechanism to harmonize learning rhythms of the individual views and cohere them to consolidate the joint representation. This mechanism is partly illustrated in Fig. 2(a) and (b), particularly from the training step 0 to 10,500. In this period, the 2D branches were converging faster than the 1D branch and were given higher weights until they started degenerating around the training step 7,500. The weights for the 2D branches were then reduced to slow down their learning while the weight for the 1D branch, which was still converging well at the time, was steadily increased to accelerate its learning. It is most likely that such a mechanism is lacking in the simple concatenation whereas late fusion of the single-view models

Table 3. Results obtained by the studied speech enhancement systems on the objective evaluation metrics.

	ESC-50	DCASE2016	GTZAN
Self-ensemble	87.35	85.38	91.10
Multi-view	86.65	85.13	90.80
Late fusion	85.80	82.05	89.70
Concat. fusion	84.44	83.85	89.00
Mel	80.15	77.18	87.30
Gammatone	76.90	77.95	86.20
CQT	58.30	75.38	83.10
Raw	75.60	57.44	81.50

is suboptimal as separate training ruled out cross-view interaction.

It should be emphasized that our primary goal in this work is to study and compare the proposed multi-view learning method to the common multi-view fusion methods with respect to a fixed network architecture rather than a comprehensive performance comparison with existing works. We, therefore, neither tailored the network architecture for the individual tasks [27, 28, 29] nor explored multi-channel combinations [30, 28, 16]. Readers should be informed that, for the databases we adopted in this study, better performance was reported in other works, such as [27, 29] for ESC-50, [16, 31] for DCASE2016, and [28] for GTZAN. Incorporating the proposed multi-view method to existing state-of-the-art networks is worth further investigation.

4. CONCLUSIONS

We presented in this work a novel multi-view learning approach for audio and music classification. The proposed multi-view network was designed to have multiple CRNN subnetworks, each handling one input view. The multi-view embedding was then produced by concatenating the embeddings learned by the single-view subnetworks. In addition to the classification branch on the multi-view embedding, the network also accommodated classification branches on the single-view subnetworks that offered a means to assess their learning behavior. Each classification branch was assigned to a weight that was adapted during training to reflect whether it is generalizing well or overfitting the data. The gradients from different classification branches were blended according to their weights for network training. In this way, different views were supposed to contribute proportionally to the multi-view embedding depending on their learning behavior. The efficacy of the proposed method was demonstrated in three different audio and music classification tasks on which the proposed method outperformed all the single-view and multi-view baselines.

5. REFERENCES

- [1] S. Pascual, M. Ravanelli, J. Serrà, A. Bonafonte, and Y. Bengio, “Learning problem-agnostic speech representations from multiple self-supervised tasks,” in *Proc. Interspeech*, pp. 161–165.
- [2] H. Phan, L. Hertel, M. Maass, P. Koch, R. Mazur, and A. Mertins, “Improved audio scene classification based on label-tree embeddings and convolutional neural networks,” *IEEE/ACM Trans. on Audio, Speech and Language Processing*, vol. 25, no. 6, pp. 1278–1290, 2017.
- [3] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, “Look, listen, and learn more: Design choices for deep audio embeddings,” in *Proc. ICASSP*, 2019.
- [4] S. S. R. Phayre, E. Benetos, and Y. Wang, “Subspectralnet - using sub-spectrogram based convolutional neural networks for acoustic scene classification,” in *Proc. ICASSP*, 2019.
- [5] H. Phan, O. Y. Chén, P. Koch, L. Pham, I. McLoughlin, A. Mertins, and M. De Vos, “Unifying isolated and overlapping audio event detection with multi-label multi-task convolutional recurrent neural networks,” in *Proc. ICASSP*, 2019.
- [6] K. J. Piczak, “Esc: Dataset for environmental sound classification,” in *Proc. ACM Multimedia*, pp. 1015–1018.
- [7] K. Choi, G. Fazekas, M. Sandler, and K. Cho, “Convolutional recurrent neural networks for music classification,” in *Proc. ICASSP*.
- [8] H. Phan, O. Y. Chén, L. Pham, P. Koch, M. De Vos, I. V. McLoughlin, and A. Mertins, “Spatio-temporal attention pooling for audio scene classification,” in *Proc. Interspeech*, 2019, pp. 3845–3849.
- [9] Z. Zhang, S. Xu, S. Zhang, T. Qiao, and S. Cao, “Learning attentive representations for environmental sound classification,” *IEEE Access*, vol. 7, pp. 130327–130339, 2019.
- [10] S. Sigtia, E. Benetos, and S. Dixon, “An end-to-end neural network for polyphonic piano music transcription,” *IEEE/ACM Transactions on Audio, Speech and Language Processing*, vol. 5, no. 24, pp. 927–939, 2016.
- [11] M. Huzaifah, “Comparison of time-frequency representations for environmental sound classification using convolutional neural networks,” *arXiv:1706.07156*, 2017.
- [12] T. Lidy and A. Schindler, “Cqt-based convolutional neural networks for audio scene classification,” in *DCASE 2016 Technical Report*, 2016, pp. 1032–1048.
- [13] W. Dai, C. Dai, S. Qu, J. Li, and S. Das, “Very deep convolutional neural networks for raw waveforms,” in *Proc. ICASSP*, 2017.
- [14] T. Harada, Y. Tokozume, Y. Ushiku, “Learning from between-class examples for deep sound recognition,” in *Proc. ICLR*, 2018.
- [15] I. McLoughlin, Z. Xie, Y. Song, H. Phan, and R. Palaniappan, “Time-frequency feature fusion for noise robust audio event classification,” *Circuits, Systems, and Signal Processing*, , no. 39, pp. 1672–1687, 2020.
- [16] L. Pham, H. Phan, T. Nguyen, R. Palaniappan, A. Mertins, and I. McLoughlin, “Robust acoustic scene classification using a multi-spectrogram encoder-decoder framework,” *Digital Signal Processing*, vol. 110, 2021.
- [17] W. Wang, D. Tran, and M. Feiszli, “What makes training multi-modal networks hard?,” in *Proc. CVPR*, 2020.
- [18] H. Phan, O. Y. Chén, P. Koch, A. Mertins, and M. De Vos, “XSleepNet: Multi-view sequential model for automatic sleep staging,” *arXiv preprint arXiv:2007.05492*, 2020.
- [19] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” in *Proc. ICML*, 2010.
- [20] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proc. ICML*, 2015, pp. 448–456.
- [21] K. Cho, B. van Merriënboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” in *Proc. EMNLP*, 2014, pp. 1724–1734.
- [22] A. Mesaros, T. Heittola, and T. Virtanen, “TUT database for acoustic scene classification and sound event detection,” in *Proc. EUSIPCO*, 2016.
- [23] G. Tzanetakis and P. Cook, “Musical genre classification of audio signals,” *IEEE Trans. on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.
- [24] C. Schoerhuber and A. Klapuri, “Constant-q transform toolbox for music processing,” in *Proc. 7th Sound and Music Computing Conference*, 2010.
- [25] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and N. Oriol, “Librosa: Audio and music signal analysis in python,” in *Proc. 14th Python in Science Conference*, 2015, pp. 18–25.
- [26] D. P. Kingma and J. L. Ba, “Adam: a method for stochastic optimization,” in *Proc. International Conference on Learning Representations (ICLR)*, 2015, pp. 1–13.
- [27] H. Wang, Y. Zou, D. Chong, and W. Wang, “Environmental sound classification with parallel temporal-spectral attention,” in *Proc. Interspeech*, 2020.
- [28] C. Liu, L. Feng, G. Liu, H. Wang, and S. Liu, “Bottom-up broadcast neural network for music genre classification,” *Pattern Recognition Letters*, 2019.
- [29] A. Guzhov, F. Raue, J. Hees, and A. Dengel, “ESResNet: Environmental sound classification based on visual domain models,” *arXiv preprint arXiv:2004.07301*, 2020.
- [30] Yoonchang Han and Kyogu Lee, “Convolutional neural network with multiple-width frequency-delta data augmentation for acoustic scene classification,” Tech. Rep., DCASE2016 Challenge, September 2016.
- [31] Y. Yin, R. R. Shah, and R. Zimmermann, “Learning and fusing multimodal deep features for acoustic scene categorization,” in *Proc. ACM Multimedia*, 2018, pp. 1892–1900.