ACOUSTIC SCENE CLASSIFICATION USING DEEP RESIDUAL NETWORKS WITH LATE FUSION OF SEPARATED HIGH AND LOW FREQUENCY PATHS

Technical Report

Mark D. McDonnell and Wei Gao

Computational Learning Systems Laboratory,
School of Information Technology and Mathematical Sciences,
University of South Australia, Mawson Lakes SA 5095, Australia

ABSTRACT

This technical report describes our approach to Tasks 1a, 1b and 1c in the 2019 DCASE acoustic scene classification challenge. Our focus was on developing strong single models, without use of any supplementary data. We investigated the use of a deep residual network applied to log-mel spectrograms complemented by log-mel deltas and delta-deltas. We designed the network to take into account that the temporal and frequency axes in spectrograms represent fundamentally different information. In particular, we used two pathways in the residual network: one for high frequencies and one for low frequencies, that were fused just two convolutional layers prior to the network output.

Index Terms— deep residual network; log-mel spectrograms; deltas and delta-deltas

1. INTRODUCTION

Task 1 in the 2019 DCASE Acoustic Scene Classification challenge required entrants to design classifiers that predicted the origin of each of many ten second recordings. The setup was similar to that in 2018 [1]. Models were trained and validated using a development set, and tested and evaluated on a leaderboard set and an evaluation set. There were ten categories (see Section 4).

Task 1 was divided into three subtasks. In subtask 1a, the development and evaluation data were both recorded by the same device, with stereo (left and right) channels, at 48 kHz. In subtask 1b ('mismatched recording devices'), the evaluation data was recorded by different devices to the development data; both were recorded using a single channel, at 44.1 kHz. In subtask 1c ('open set'), an additional 11-th class representing "unknown" was included in the data. Like subtask 1b, subtask 1c used single channel recordings at 44.1 kHz, but unlike subtask 1b, recordings were all collected from the same device used in development set.

Code for training our models and running trained models is at `https://github.com/McDonnell-Lab/DCASE2019-Task1` .

2. STRATEGY AND MODEL INNOVATIONS

Highest accuracy in previous approaches to scene classification have arisen from treating spectrograms of acoustic scenes as though they are images, and training deep Convolutional Neural Networks (CNNs) on these spectrograms using best practise image classification methods, e.g. [2–4]. We also adopt the use of a CNN applied to spectrograms, but aim to improve on previous designs in several ways described in this Section. We did not use any additional data to that provided by the challenge organizers.

2.1. Acoustic feature extraction

Past entries into DCASE challenges have used a range of approaches to forming image-like spectrograms for CNN processing. These have included log-mel spectrograms, MFCCs, perceptual weighted power spectrograms, CQT spectrograms and so forth.

Our approach was inspired by past work on automated phoneme recognition using a CNN [5], that used log-mel energies, and additionally calculated deltas and delta-deltas from these, i.e. approximations to the first and second temporal derivatives of the spectrum.

For Tasks 1b and 1c, with raw data in mono, we therefore had 3 input channels to our CNN. For Task 1a, where there were left and right channels, we calculated log-mel spectrograms and deltas and delta-deltas for both, and consequently, the overall input to our CNN had 6 channels. We did not attempt to add or subtract left and right channels, since for an experiment without deltas and delta-deltas we achieved almost identical results when using the left and right channels as two independent CNN channels, compared with instead adding and subtracting them. We also found no benefits from decomposition into harmonic and percussive (HPSS) components as used by some previous DCASE contest submissions [2].

Compared to log-mel spectrograms as the only input channels, we observed improved error rates on the official DCASE 2019 development validation split when using deltas and delta-deltas (see Table 1).

2.2. CNN design

We note that spectrograms have characteristics different from images [6]. For example, one object placed behind another is entirely occluded in a photograph, whereas sounds from two sources superimpose such that frequency features in a spectrogram can arise from a combination of the two sources. Another important difference is that an object can appear anywhere in an image, and carry the same meaning, whereas patterns of features at low frequencies may represent different physical origins from those at higher frequencies. Consequently the time and frequency axes that comprise the two axes of a spectrogram are not of the same nature as the two spatial axes in an image. Thirdly, frequency features at any point in time can be non-local, due, for example, to harmonics.

The second and third point leads us to design a CNN in which the frequency and time axes are treated differently. The main deviations from a standard deep CNNs are described in Section 3.

2.3. Aggressive regularization and data augmentation

We found significant levels of overfitting, i.e. the training loss and error rate for our trained models applied to the training set were close to zero for sufficiently large models. Therefore, we used several forms of regularization and data augmentation.

Like many entries in previous DCASE challenges, we used mixup augmentation [7], and like most deep CNNs for image classification, we used weight decay for all convolutional layers. We also experimented with shift-and-crop augmentation, but found best results when only relatively mild temporal cropping was used. Finally, we made use of a new approach from image classification which is not in common practice, which was to add a form of regularization where batch normalization layers did not have their offset and scale parameters learned [8].

Coupled with using these approaches, we found it helpful to train for a very large number of epochs (we used 510) in a warm-restart learning-rate schedule [9].

3. METHODS

We used the same network architecture and training approach for all of Tasks 1a, 1b and 1c, except for adjustments for the slower sample rate in a single channel for Tasks 1b and 1c.

Details are as follows. All networks were trained using keras (version 2.2.4) with a tensorflow (version 1.12.0) backend.

3.1. Acoustic file preprocessing

To calculate our log-mel energies, we used 2048 FFT points, the original sampling rate of the acoustic files (48 KHz for Task 1a, 44.1 KHz for Tasks 1b and 1c), frequencies from 0 to half of the sampling rate, a hop-length of 1024 samples, and the HTK formula to define the mel scale [10]. Our implementation used python, and the LibROSA library[1]. Our resulting spectrograms were of size 469 (Task 1a) or 431 (Tasks 1b and 1c) time samples, and 128 frequency bins. We calculated the log-mel deltas and delta-deltas without padding, which reduced the number of time samples to 461 (Task 1a) or 423 (Tasks 1b and 1c).

3.2. Splitting of high and low frequencies

The CNN we designed has two mostly parallel paths that combine only using late fusion by concatenation of frequency axes, two convolutional layers before the network output. The overall network input has 128 frequency dimensions, but these are immediately split in the network such that dimensions 0 to 63 is processed by a residual network with 17 convolutional and dimensions 64 to 127 by another. All kernels in these paths are 3×3. After these stacks, each pathway is concatenated to form 128 frequency dimensions, and then operated on by two 1×1 convolutional layers. The second of these layers reduces to the number of classes (10 for Task 1a and 1b, and 11 for Task 1c). This is followed by a batch normalization layer, a global average pooling layer, and softmax.

The idea with the two branches is that the frequency features to be learned for high frequencies are likely to different to those for low frequencies. Therefore, we hypothesise that better learning will occur if convolutional kernels do not get get applied at all frequencies in a spectrogram.

[1]`https://librosa.github.io/librosa/`

The final two 1×1 layers effectively act as a two layer non-convolutional neural network that weights the contributions of each channel in each branch for classification of the scene at each frequency. In the temporal axis, the global average pooling layer works like in a deep CNN for image classification: it equally weights many globally processed "views" of the image. We discuss the frequency axis in the next subsection.

3.3. No downsampling in frequency layers

The input to our network for training has 400 time samples (due to random temporal cropping—see below) and 128 frequencies. Due to the all-convolutional nature of the network, at inference time we can use larger number of time samples, and use all 461 (Task 1a) or 431 (Tasks 1b and 1c) samples provided by our audio preprocessing.

In order to ensure the CNN can learn global temporal information across all time samples, we use standard image classification practise of regularly downsampling in time using stride 2 convolutional layers. The principle is that an important cue could happen with equal likelihood at any point in time in a 10 second sample, just like objects in images can appear in any spatial location.

However, in the frequency axis we do not downsample. Consequently, the number of frequency dimensions in the feature maps for each path remains constant at 64 throughout the network. Hence, at the point where the two branches are concatenated, each path has processed a frequency-axis receptive field of 35 dimensions.

Consequently, the global average pooling layer does not merge equal global "views" in the frequency axis, but instead averages over different overlapping views spanning 35 dimensions, rather than global views. We therefore investigated using a final layer with learned weights for merging each frequency dimension, but found better results wit global average pooling.

3.4. Other CNN design aspects

The residual network design is a pre-activation variety [8, 11], where the input to each convolutional layer first is processed by a batch normalization layer and then a ReLU activation. In the residual paths, when the number of channels needs to be increased before summation of different paths, we used zero padding in the channel dimension as in [8], rather than 1×1 convolutions.

Using a technique introduced in [8], the very first layer of our network was a batch normalization layer with learned offset and scale parameters. This enabled us to avoid assumed forms of normalization of the features passed into the network.

Overall, our networks had approximately 3.2 million trainable parameters.

3.5. Regularization and data augmentation

We used the following:

- **weight decay**: we used an aggressively large value of 5×10^{-4} (i.e. 1×10^{-3} when set in keras) on all convolutional layers.
- **Not learning batch normalization scale and offset**: it was shown in [8] that for datasets and networks with significant overfitting, turning off learning of batch normalization scale and offset (except in the very first layer) has a regularization effect resulting in improved test error rates on the CIFAR-100 benchmark. We used this approach here.
- **Mixup and temporal crop augmentation:** As found by others in past DCASE challenges, we found it very useful to use

mixup augmentation, using the same approach as [2], with $\alpha = 0.2$. We additionally used crop augmentation in the temporal axis: each of the two samples combined using mixup were first cropped independently and randomly from 461 (Task 1a) or 423 (Tasks 1b and 1c) dimensions down to 400.

3.6. Training

We used backpropagation and stochastic gradient descent, with a batch size of 32, momentum of 0.9, and the cross-entropy loss function. Each network was trained for 510 epochs using a warm restart learning rate schedule that resets the learning rate to its maximum value of 0.1 after $2, 6, 14, 30, 126$ and 254 epochs, and then decays according to a cosine pattern to 1×10^{-5}. It was shown by [9] and verified by [8] that this approach can provide improvements in accuracy on image classification relative to using stepped schedules.

3.7. Inference

Due to their all convolutional nature, spectrograms of arbitrary duration can be processed by our CNNs. At inference time, full-duration spectrograms were operated on by our trained CNNs.

We did not use any inference-time processing in Tasks 1a and 1b. For task 1c, the ranking metric for the DCASE 2019 Challenge was weighted the accuracy on known classes and the accuracy on 'unknown' data equally. Consequently, at inference time we weighted the softmax output for the unknown class by a factor of 5 (determined using the development set validation data) before applying the argmax operator to select the predicted class.

Past DCASE Challenges (and other machine learning contests) tend to be won by ensembles combined using either simple averaging, or by meta-learning approaches involving stacking. We did not concentrate our efforts on this aspect, instead preferring to seek the best single network we could. We investigated only simple ensembling by averaging the softmax output of models trained on all development set data. We typically observed less than 1% improvement on raw accuracy.

3.8. Validation

An official train/validation split of the DCASE development data was provided for each subtask, roughly in a 70:30 ratio. We designed and selected models using these splits and then retrained each model using the entirety of the development data before running the models on leaderboard or evaluation data for submission.

4. RESULTS

This section contains results on the official contest validation splits of the Task 1a, 1b and 1c development sets.

The DCASE 2019 Task 1 challenge is evaluated using accuracy calculated as the average of the class-wise accuracy, also known as 'balanced accuracy.' Given the development set validation split has unequal numbers within each class, this means balanced accuracy is not exactly equal to the raw classification accuracy. However, to indicate the value of using log-mel deltas and delta-deltas, Table 1 shows the raw accuracy for each task.

The per-class precision and recall, and the average over each class, are shown in Tables 2, 3 and 4 for Tasks 1a, 1b and 1c respectively. Confusion matrices are shown in Figures 1, 2 and 3.

Table 1: Raw accuracies.

Task	No deltas	Best single model
1a	81.4%	82.3%
1b (device A)	78.5%	80.0%
1b (device B and C)	62.5%	66.3%
1c (known classes)	82.3%	76.8%
1c (unknown)	59.0%	63.9%

5. DISCUSSION

For each of Tasks 1a, 1b and 1c, four submissions were permitted. We submitted results for two single models, an ensemble formed by averaging the raw predictions of these, and an ensemble formed by averaging two independently trained copies of our best model.

5.1. Remarks on Task 1a

We observe that the per-class precision and recall for our best model had no greater than a gap of 0.14, indicating the model is well-balanced across all classes. Public square has the worst recall, and shopping mall the worst precision.

5.2. Remarks on Task 1b

Our best model generalized to devices B and C well on some classes but very poorly on some others. The main confusion cases are airport being classified as shopping mall, tram as metro station and public square as street traffic. In future work, inference-time weightings for classes with low recall might enhance performance.

5.3. Remarks on Task 1c

The evaluation metric in Task 1c weights the accuracy (recall) for the unknown class equally with the balanced accuracy for the ten known classes. This encourages models that err on the side of predicting unknown, which for us resulted in a low precision for the unknown class, and a relatively low recall for each of the known classes, as shown in Table 4, and Figure 3.

Table 2: Task 1a, best single model.

Class	Recall	Precision
airport	0.74	0.81
bus	0.89	0.87
metro	0.82	0.82
metro station	0.81	0.82
park	0.91	0.92
public square	0.70	0.80
shopping mall	0.81	0.77
street pedestrian	0.82	0.84
street traffic	0.94	0.80
tram	0.81	0.79
Average	82.3%	82.4%

Table 3: Task 1b, best single model, devices B and C only.

Class	Recall	Precision
airport	0.36	0.68
bus	0.86	0.72
metro	0.77	0.55
metro station	0.79	0.63
park	0.86	0.94
public square	0.23	0.66
shopping mall	0.79	0.51
street pedestrian	0.55	0.57
street traffic	0.95	0.73
tram	0.47	0.88
Average	66.3%	68.6%

Table 4: Task 1c, best single model.

Class	Recall	Precision
airport	0.43	0.88
bus	0.76	0.94
metro	0.54	0.91
metro station	0.60	0.96
park	0.80	0.94
public square	0.50	0.77
shopping mall	0.61	0.84
street pedestrian	0.47	0.73
street traffic	0.87	0.90
tram	0.63	0.84
Known class average	62.0%	87.0%
unknown	80.3%	17.6%
Overall average	63.7%	80.7%
Evaluation score	71.1%	N/A

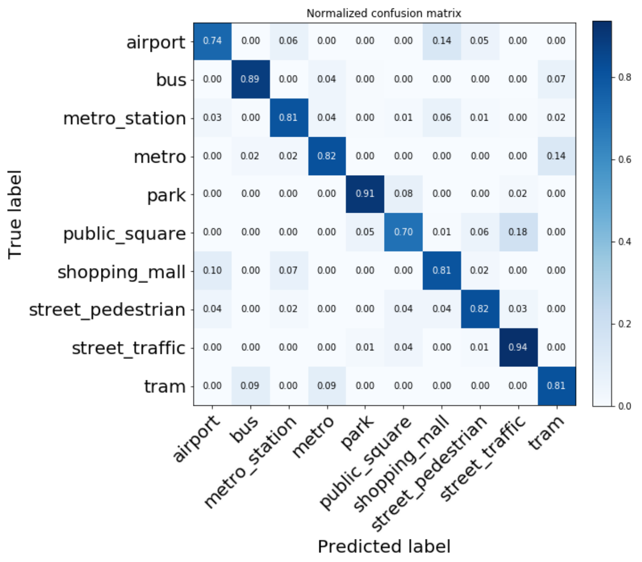

Figure 1: Normalized confusion matrix for Task 1a, best single model (normalization using the true label, i.e. recall mode).

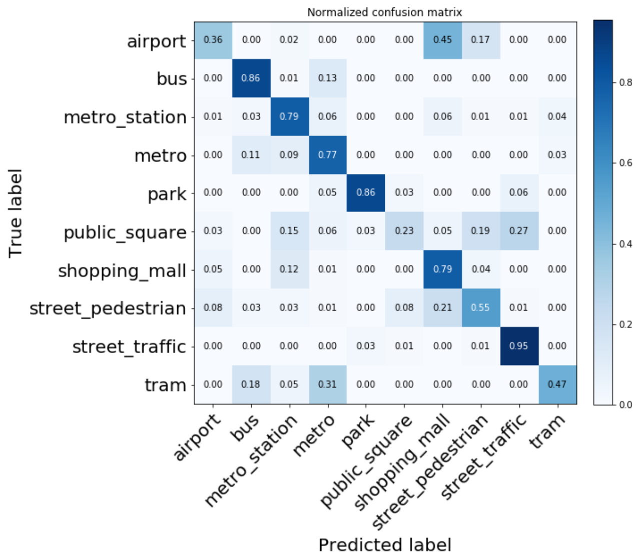

Figure 2: Normalized confusion matrix for Task 1b, best single model (normalization using the true label, i.e. recall mode), for devices B and C only.

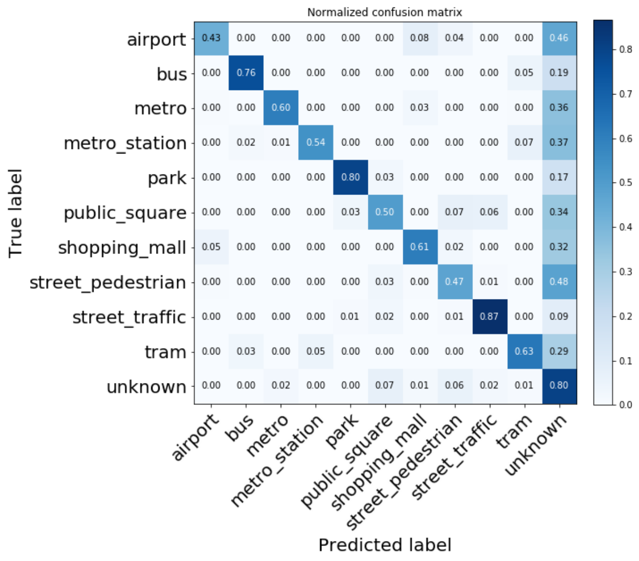

Figure 3: Normalized confusion matrix for Task 1c, best single model (normalization using the true label, i.e. recall mode).

References

[1] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, November 2018, pp. 9–13. [Online]. Available: https://arxiv.org/abs/1807.09840

[2] Y. Sakashita and M. Aono, "Acoustic scene classification by ensemble of spectrograms based on adaptive temporal divisions," Tech. Rep., 2018, DCASE 2018 technical reports.

[3] H. Zeinali, L. Burget, and J. H. Cernocky, "Convolutional neural networks and x-vector embedding for DCASE2018 acoustic scene classification challenge," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, November 2018, pp. 202–206.

[4] M. Dorfer and G. Widmer, "Training general-purpose audio tagging networks with noisy labels and iterative self-verification," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, November 2018, pp. 178–182.

[5] Y. Zhang, M. Pezeshki, P. Brakel, S. Zhang, C. Laurent, Y. Bengio, and A. Courville, "Towards end-to-end speech recognition with deep convolutional neural networks," in *Interspeech*, 2016, pp. 410–414.

[6] D.Rothman, "What's wrong with CNNs and spectrograms for audio processing?" Tech. Rep., 2018, https://towardsdatascience.com/whats-wrong-with-spectrograms-and-cnns-for-audio-processing-311377d7ccd.

[7] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018.

[8] M. D. McDonnell, "Training wide residual networks for deployment using a single bit for each weight," 2018, in Proc. ICLR 2018; arxiv: 1802.08530.

[9] I. Loshchilov and F. Hutter, "SGDR: stochastic gradient descent with restarts," *CoRR*, vol. abs/1608.03983, 2016. [Online]. Available: http://arxiv.org/abs/1608.03983

[10] S. J. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK Book Version 3.4*. Cambridge University Press, 2006.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," Microsoft Research, Tech. Rep., 2016, arxiv.1603.05027.