PARTIALLY-SHARED CONVOLUTIONAL NEURAL NETWORK FOR CLASSIFICATION OF MULTI-CHANNEL RECORDED AUDIO SIGNALS

Technical Report

Kazuhiro Nakadai and Danilo R. Onishi

Honda Research Institute Japan, 8-1 Honcho, Wako, Saitama, JAPAN,
{nakadai,d.onishi}@honda-ri.com

ABSTRACT

This technical paper presents the system used in our submission for task 5 of the DCASE 2018 challenge [1]. We proposed a partially-shared convolutional neural network, which is a multi-task system that contains a common input (the multi-channel log Mel features) and two output branches, a classification branch, which outputs the predicted class, and a regression branch, which outputs a single-channel representation of the multi-channel input data. Since the system has a shared network between classification and regression, training for regression is expected to enhance another training for classification and vice versa. Because task 5 aims at classification based on multi-channel audio input, we tried to improve classification performance with this system by training classification and regression together. By applying the proposed system incorporated with parameter tuning of the baseline CNN system, we confirmed that the classification F1 score increased to 89.94% in four-fold cross validation, while the baseline system achieved 84.50% .

Index Terms— CNN, sound source classification, sound source separation, multi-task training

1. INTRODUCTION

The system proposed for DCASE 2018 Challenge task 5 is a multi-task CNN which contains a classification branch and a regression branch. Furthermore, in the first few layers of the network some convolutional filters are shared among both paths. The classification path is trained with a cross-entropy loss function with the labeled classes as targets, while the regression path is trained with a mean-squared-error loss function and the training targets are pre-computed single-channel representations of the multi-channel input data. Multi-task training with shared filters allows the shared weights to train on data from both tasks. Furthermore, by applying weighting factors to the classification and regression losses after the forward pass and prior to back-propagation, one task can act as a regularizer to the other and thus prevent overfitting. The partially-shared architecture was initally proposed as a way to harness unlabeled data, so that even if the classification branch is trained on a small labeled data set, using a larger regression data set helps the shared filter to achieve weights that would improve the classification side. However, since task 5 of the DCASE 2018 Challenge has a considerable amount of data, the proposed architecture will be used to further boost the classification performance.

2. PROPOSED METHOD

The partially-shared architecture initially proposed in [3] used fully-connected layers as building blocks. This report extends the concept by using convolutional layers instead.

Figure 1 shows PS-CNN, which is an application of multi-tasking or, as we call partially-shared architecture, with CNN. In normal CNN for color image classification, an input to the network is a set of three two-dimensional images, and each image corresponds to R, G, or B color channels. In order to apply this CNN to audio data, a multichannel audio input to this network is converted into two-dimensional acoustic feature matrices (images). Each matrix is generated by temporally aligning acoustic feature vectors for each channel. Therefore, the number of images is the same as that of microphones (channels) in the microphone array.

In this implementation of PS-CNN, the acoustic feature matrices for selected channels are shared, and the others are not, that is, the shared architecture is applied per channel. This achieves two kinds of training, that is, for regression and classification in PS-CNN. In a pooling layer, a pooling process is performed per channel in a similar way to conventional CNN.

Let us formulate processing in a convolution layer, which is unique in this network. An input to the j-th channel in the i-th convolution layer is defined by

$$x^i_j = [x^i_{1,1,j}, \cdots, x^i_{m,n,j}, \cdots, x^i_{M^i,N^i,j}], \tag{1}$$

where M^i and N^i define the size of one two-dimensional data in the i-th convolution layer.

PSA in Equation (1) is defined as follows: The j-th channel belongs to subnetwork 1 when $1 \leq j \leq C^i_1$, and it belongs to subnetwork 2 when $C^i_2 \leq j \leq C^i$, where C^i is the number of channels in the i-th convolution layer. This means that the j-th channel is shared by both subnetworks when $C^i_2 \leq j \leq C^i_1$.

When the k-th filter in the i-th convolution layer is defined by

$$w^i_k = [w^i_{1,1,k}, \cdots, w^i_{m,n,k}, \cdots, w^i_{V^i,H^i,k}], \tag{2}$$

where V^i and H^i define the size of the filter in the i-th convolution layer.

The output of the j-th channel in the i-th layer is defined by,

Table 1: PS-CNN structure. Values separated by slashes are given for classification branch, shared branch and regression branch, in this order. When a single value is given, it is used in all branches.

Parameter	CNN	CNN+shared	PS-CNN
conv 1 #filters	32 / 0 / 0	32 / 32 / 0	32 / 32 / 32
conv1 size (time x freq)	5×5	5×5	5×5
maxpool1 (time x freq)	5×1 / – / –	5×1 / 5×1 / –	5×1 / 5×1 / 5×1
dropout1	0.2 / 0 / 0	0.2 / 0 / 0	0.2 / 0 / 0
conv 2 #filters	64 / 0 / 0	64 / 64 / 0	64 / 64 / 64
conv2 size (time x freq)	3×1	3×1	3×1
maxpool2 (time x freq)	– / – / –	– / – / –	– / – / –
dropout2	0.2 / 0 / 0	0.2 / 0 / 0	0.2 / 0 / 0
conv 3 #filters	64 / 0 / 0	64 / 0 / 0	64 / 0 / 64
conv3 size (time x freq)	3×1	3×1	3×1
maxpool3 (time x freq)	199×2 / – / –	199×2 / – / –	199×2 / – / –
dropout3	0.2 / 0 / 0	0.2 / 0 / 0	0.2 / 0 / 0
conv 4 #filters	64 / 0 / 0	64 / 0 / 0	64 / 0 / 1
conv4 size (time x freq)	1×1	1×1	1×1
maxpool4 (time x freq)	– / – / –	– / – / –	– / – / –
dropout4	0.2 / 0 / 0	0.2 / 0 / 0	0.2 / 0 / 0
fully-conn1 #nodes	64 / 0 / 0	64 / 0 / 0	64 / 0 / 64
dropout5	0.2 / 0 / 0	0.2 / 0 / 0	0.2 / 0 / 0
output #nodes	9 / 0 / 0	9 / 0 / 0	9 / 0 / 39880

Table 2: Per-class and total F1 scores

class	Baseline [2]	CNN	CNN + shared	PS-CNN
Absence	85.41	91.17	89.47	91.14
Cooking	95.14	96.02	95.81	96.10
Dishwashing	76.73	81.89	82.16	83.66
Eating	83.64	91.95	92.61	91.55
Other	44.76	58.61	60.13	61.79
Social activity	93.92	95.98	96.30	96.25
Vacuum cleaning	99.31	99.01	99.89	99.90
Watching TV	99.59	99.82	99.73	99.80
Working	82.03	88.80	88.25	89.29
Average	84.50	89.25	89.37	**89.94**

$$y^i_{m,n,j'} = \begin{cases} \sigma\left(\sum_{j=1}^{C_1^i}\sum_{v=1}^{V^i}\sum_{h=1}^{H^i} w_{v,h,j}x^i_{m+v,n+h,j} + b^i_j\right) \\ \forall(1 \le j' \le C_2^{i+1}) \\ \sigma\left(\sum_{j=C_2^i}^{C_1^i}\sum_{v=1}^{V^i}\sum_{h=1}^{H^i} w_{v,h,j}x^i_{m+v,n+h,j} + b^i_j\right) \\ \forall(C_2^{i+1} \le j' \le C_1^{i+1}) \\ \sigma\left(\sum_{j=C_2^i}^{C^i}\sum_{v=1}^{V^i}\sum_{h=1}^{H^i} w_{v,h,j}x^i_{m+v,n+h,j} + b^i_j\right) \\ \forall(C_1^{i+1} \le j' \le C^{i+1}) \end{cases} \quad (3)$$

where b^i_j is a bias for the j-th channel in the i-th layer, and $\sigma(\cdot)$ is an activate function. For our setting, ELU [4] was used.

Regression data is generated by applying a sound source separation method to the original multi-channel data. In this case, Geometric High-order Decorrelation-based Source Separation with Adaptive Step-size control (GHDSS-AS) [5] was used. Note that, since it's a regression task, labels are not required, and therefore target data can be generated even from unlabeled multi-channel audio.

2.1. Sound Source Separation for Generating Target Data

GHDSS-AS is one of the hybrid algorithms of beamforming and blind separation such as GSS proposed by Parra *et al.* [6] and its online algorithm developed by Valin *et al.*[7]. GHDSS-AS has two extensions from GSS-based algorithms. One is the cost function for separation. GHDSS-AS uses higher order correlation as a cost function, while GSS uses the second order cross power correlation. As used in many independent component analysis algorithms, high order correlation shows higher separation performance, and thus,

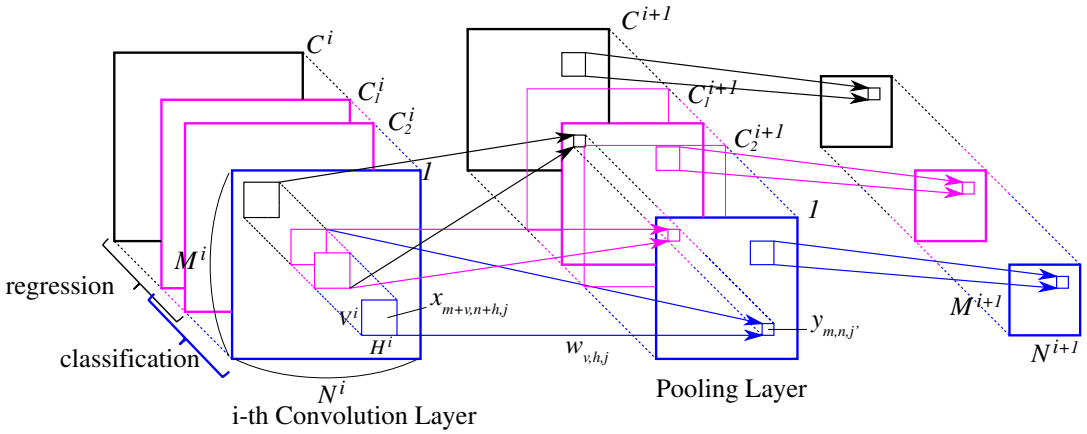

Figure 1: A convolution-pooling layer in PS-CNN.

GHDSS-AS fits the task 5 of the DCASE 2018 challenge. The other extension is an adaptive step-size control, which provides faster adaptation using stochastic gradient and shorter time frame estimation.

Using the multi-channel input signal $\mathbf{x}_t$ and the separated signal $\mathbf{y}_t$ at the t-th frame in frequency domain, GHDSS-AS can be formulated as,

$$\mathbf{y}_t = \mathbf{W}_t \mathbf{x}_t, \tag{4}$$

$$\mathbf{W}_t = \mathbf{W}_{t-1} - \mu_{S_t} \mathbf{J}'_S(\mathbf{W}_{t-1}) + \mu_{G_t} \mathbf{J}'_G(\mathbf{W}_{t-1}), \tag{5}$$

$$\mu_{S_t} = \frac{\|\phi(\mathbf{y}_{t-1})\mathbf{y}_{t-1}^H - \mathrm{diag}[\phi(\mathbf{y}_{t-1})\mathbf{y}_{t-1}^H]\|^2}{8\|\phi(\mathbf{y}_{t-1})\mathbf{y}_{t-1}^H - \mathrm{diag}[\phi(\mathbf{y}_{t-1})\mathbf{y}_{t-1}^H]\tilde{\phi}(\mathbf{y}_{t-1})\mathbf{x}_{t-1}^H\|^2}, \tag{6}$$

$$\mu_{G_t} = \frac{\|\mathrm{diag}[\mathbf{W}_{t-1}\mathbf{D} - \mathbf{I}]\|^2}{8\|\mathrm{diag}[\mathbf{W}_{t-1}\mathbf{D} - \mathbf{I}]\mathbf{D}^H\|^2}, \tag{7}$$

where $\mathbf{J}'_S(\mathbf{W}_{t-1})$ and $\mathbf{J}'_G(\mathbf{W}_{t-1})$ are complex gradients [8] of $J_S(\mathbf{W}_{t-1})$ and $J_G(\mathbf{W}_{t-1})$, which are defined by,

$$J_S(\mathbf{W}_{t-1}) = \|\phi(\mathbf{y}_{t-1})\mathbf{y}_{t-1}^H - \mathrm{diag}[\phi(\mathbf{y}_{t-1})\mathbf{y}_{t-1}^H]\|^2, \tag{8}$$

$$J_G(\mathbf{W}_{t-1}) = \|\mathrm{diag}[\mathbf{W}_{t-1}\mathbf{D} - \mathbf{I}]\|^2, \tag{9}$$

where $\mathbf{D}$ represents the transfer function between the microphone array and sound sources, $\|\cdot\|^2$ indicates the Frobenius norm, $\mathrm{diag}[\cdot]$ is the diagonal operator, and H represents the conjugate transpose operator. For a nonlinear function, $\phi(\mathbf{y})$, hyperbolic-tangent-based function [9] was selected.

Eqs. (8) and (9) show the cost functions used in GHDSS-AS, and Eqs. (6) and (7) show the adaptive step-size control. From Eqs. (4) – (9), it is obvious that all processes of GHDSS-AS can be calcurated from $\mathbf{D}$ and the results in the previous frame and the transfer function. This means that GHDSS-AS is almost parameterless besides $\mathbf{D}$.

$\mathbf{D}$ can be obtained either from actual impulse response measurements (more specifically, time-stretched pulses), or by numerical simulation using the geometric relationship between the microphones in the array and the sound sources. The former method usually performs better than the latter, since the geometric-calculation-based method excludes reflected sounds and, therefore, does not take into account the distortions due to the environment. However, since impulse response measurements were not available for the DCASE 2018 challenge task 5, the latter method was used.

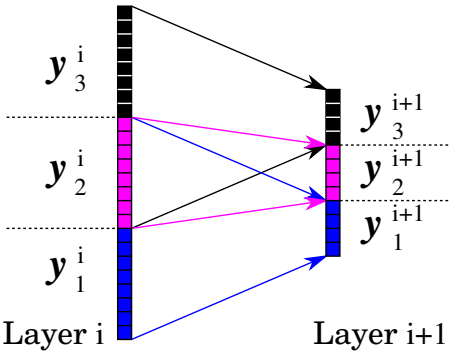

Figure 2: Concatenation of layers in the partially-shared architecture. y_1 is the classification branch, y_2 the shared branch, and y_3 the regression branch.

The center of the linear microphone array used in the task measurements was set as the origin of the coordinate system for the geometric positions, and then virtual sound sources located at a radius of 1.0 meter from the origin were placed with an azimuth interval of five degrees. Also, since the microphone arrays were placed on the walls, only the front plane is used, and therefore $180°/5° = 36$ sound sources were simulated. Finally, the impulse responses simulated geometrically from the microphones and sources positions were used to generate the transfer function $\mathbf{D}$.

3. EXPERIMENT SETTINGS

For a sound source separation algorithm, we used the GHDSS-AS implementation from the open source software for robot audition *Honda Research Institute Japan Audition for Robots with Kyoto University* (HARK) [10]. GHDSS-AS is reported as one of the best microphone array separation algorithms [11].

For the acoustic features used as the input of the networks, we adopted Mel filter bank features. The acoustic signals in the datasets were sampled at 16 kHz and 16 bit. We framed them with a frame width of 512 samples (32 ms) and a frame shift of 160 samples (10 ms). We used a complex window [10] as the window function when performing Short-Time Fourier Transform (STFT). From a

complex spectrum obtained by STFT, we calculated 40-dimensional Mel filter bank feature vectors by setting a lower and higher cut-off frequency to be 63 and 8,000 Hz, respectively.

Input vectors have 4 channels, 997 time frames (slightly down from 1,000 frames due to preprocessing) and 40 frequency features. Labels for the regression branch have similar dimensions, except they are single-channel vectors.

The network settings are shown in Table 1. $Adam$ [12] was chosen as the optimizer with a learning rate of 0.001, and the batch size was 64. A dropout [13] factor of 0.2 was used in the classification-only branch for both convolutional and fully-connected layers. An L2 penalty factor of 0.0001 is applied to all trainable weights, and the ELU activation function [4] was used. Also, prior to summing up the classification cross entropy loss and the regression mean square error (MSE), a multiplying factor of 10.0 is applied to the MSE. We performed four-fold cross-validation using the folds defined by the DCASE challenge. The number of output nodes in the classification branch matches the number of classes (9) of the task, and the output of the regression branch is set to match the size of the separated Mel filter bank features (1 channel x 40 features x 997 frames = 39880). The number of epochs was fixed at 100, and the model with the best cross-validation F1 score for each fold was used for evaluation.

4. RESULTS

Table 2 shows the macro-averaged F1 scores, averaged over the four folds. Each model structure is described in Table 1. *CNN* refers to the model with only plain convolutional layers (the classification branch of the *PS-CNN* model). *CNN + shared* is the model with classification branch plus the shared branch, thus each partially-shared layer is similar to Figure 2 without the y_3 branch. Since this version has no regression branch, only classification is performed. *PS-CNN* is the full model with classification, shared and regression branches.

By tuning our *CNN* architecture, the error was reduced by 30.6% relative to the provided baseline [2]. This can likely be attributed to the deeper architecture (the baseline contains only two convolutional layers as opposed to four layers in ours) and the use of all the audio channels in the input (the baseline inputs each channel of the audio data separately).

The *CNN + shared* model performed similarly to *CNN*, so adding the shared layers does not improve over a plain architecture, but it does not hurt the performance either. Finally, the full *PS-CNN* architecture had a further 6.4% relative error reduction over the tuned *CNN*. This result shows that adding the regression branch to the intermediate model influences the shared layers in a way that benefits the classification.

5. CONCLUSION

This report proposed the use of PS-CNN, a multi-task CNN architecture with classification-specific and regression-specific branches, along with filters that are shared between them, for the 5th task of the DCASE 2018 Challenge. The labels for the regression branch were generated by applying the GHDSS-AS sound source separation technique to the provided multi-channel data, and the transfer function required by the method was generated by geometric simulation from the provided microphone array information. Our tuned base CNN model achieved a 30.6% error decrease relative to the baseline provided by the challenge, and our proposed architecture had a further 6.4% relative error reduction over the basic classification-only CNN with a similar structure. Future topics for study and improvement include trying a deeper architecture and tuning the separation algorithm parameters.

6. REFERENCES

[1] http://dcase.community/challenge2018/.

[2] http://dcase.community/challenge2018/task-monitoring-domestic-activities.

[3] T. Morito, O. Sugiyama, R. Kojima, and K. Nakadai, "Partially shared deep neural network in sound source separation and identification using a uav-embedded microphone array," in *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1299–1304.

[4] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *CoRR*, vol. abs/1511.07289, 2015. [Online]. Available: http://arxiv.org/abs/1511.07289

[5] H. Nakajima, K. Nakadai, Y. Hasegawa, and H. Tsujino, "Correlation matrix estimation by an optimally controlled recursive average method and its application to blind source separation," *Acoustical Science and Technology*, vol. 31, no. 3, pp. 205–212, 2010.

[6] L. C. Parra and C. V. Alvino, "Geometric source separation: Mergin convolutive source separation with geometric beamforming," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 6, pp. 352–362, 2002.

[7] J.-M. Valin, S. Yamamoto, J. Rouat, F. Michaud, K. Nakadai, and H. G. Okuno, "Robust recognition of simultaneous speech by a mobile robot," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 742–752, 2007.

[8] D. Brandwood, "A complex gradient operator and its application in adaptive array theory," *IEE Proc.*, vol. 130, no. 1, pp. 251–276, 1983.

[9] H. Sawada, R. Mukai, S. Araki, and S. Makino, "Polar coordinate based nonlinear function for frequency-domain blind source separation," in *2002 IEEE Int' l. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2002)*, 2002, pp. 1001–1004.

[10] K. Nakadai, H. G. Okuno, and T. Mizumoto, "Development, deployment and applications of robot audition open source software HARK," *Journal of Robotics and Mechatronics*, vol. 29, no. 1, pp. 16–25, 2017.

[11] H. G. Okuno and K. Nakadai, "Robot audition: Its rise and perspectives," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2015)*, 2015, pp. 5610–5614.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[13] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.