USING ARCFACE METRIC LEARNING FOR LOW-COMPLEXITY ACOUSTIC SCENE CLASSIFICATION

Technical Report

Kristóf Horváth[§]*, Harsh Purohit, Ryo Tanabe, Kota Dohi, Takashi Endo, Masaaki Yamamoto, Tomoya Nishida, Yohei Kawaguchi*

horvath.kristof.sz@gmail.com,
[harsh_pramodbhai.purohit.yf, yohei.kawaguchi.xk]@hitachi.com

Hitachi Ltd.

ABSTRACT

In this technical report we present our submissions for DCASE 2021 Challenge Task 1A. For the low-complexity model, we used both a MobileNetV2-based model and a ResNet-based model with reduced number of layers and trained it using ArcFace metric learning. To increase the accuracy, we used test-time augmentation (TTA) during inference. On the development dataset, our models attain an ASC accuracy of around 54–55%, while having less than 128 kB of total parameters[1].

***Index Terms*—** acoustic scene classification, MobileNetV2, ResNet, ArcFace, test-time augmentation

1. INTRODUCTION

Acoustic scene classification (ASC) is a problem where the goal is to identify the environment using sound recordings and assign a label from a set of given classes. ASC has been a popular topic for decades, and the Detection and Classification of Acoustic Scenes and Events (DCASE) community provides a challenge to encourage sound scene research.

There are two different subtasks for DCASE 2021 challenge Task 1. We focused only on subtask A, which goal is to give a low-complexity, ten-class ASC system with size limit of 128 kB (131072 bytes) of trainable, nonzero parameters [1].

The development dataset for the task, named *TAU Urban Acoustic Scenes 2020 Mobile development dataset* contains 64 hours of recordings from 10 European cities in 10 different acoustic scenes [2]. The samples were recorded using multiple devices, including 3 real devices (marked as A, B, C). Additionally, synthetic data for 6 simulated devices (S1–S6) was created based on the original recordings. The evaluation dataset contains additional samples from 2 more European cities, one new real device (marked as D) and 3 new simulated devices (S7–S9).

The development dataset is provided with a training/test split. 70% of the samples from each device is included in the training set, and 30% is allocated for testing. Some devices (S4–S6) appear only in the test subset. The training set is already balanced by the organizers in terms of scene classes and recording devices.

In our submission we used TensorFlow 2.1.0 to implement the neural network [3].

[§]Intern student

[1]Source code: `https://github.com/hekkelek/DCASE2021_task1a`

2. FEATURE EXTRACTION

The DCASE 2021 dataset consists of 10 second long recordings of 10 different acoustic scenes, recorded with 44.1 kHz sampling frequency. As a feature extraction layer, we used a log-mel filter bank with 128 Mel frequency bins. The recordings are transformed using a 2048 sample long short-time Fourier transform (STFT) window with frame shift of 1024 samples. The spectra are then mapped to the Mel scale using the HTK formula, and the natural logarithm of each value is calculated. After this, the features are scaled to $[0, 1]$. After the scaling, features are decomposed to harmonic and percussive components, and concatenated to the Mel spectrogram as new channels. Thus, the input tensor size is $128 \times 431 \times 3$.

The feature extraction was implemented using the `librosa` library [4].

Because the overwhelming majority of samples were recorded using device A, we applied additional balancing to the train split of the dataset. We also noticed that the recordings were imbalanced not only in terms of devices, but in terms of cities too. Therefore, in order to balance, we used subsampling on the devices and cities: at least 7 samples were recorded in each city using all of the devices, so we randomly sampled 7 recordings from 10 cities, 6 devices and 10 scenes, which resulted in 4200 training samples.

3. AUGMENTATION

In order to reduce overfitting, several augmentation methods were used. These can be categorized as time-domain and feature-domain augmentations.

The following list contains the time-domain approaches. These operate on the audio recordings and generate additional features in our experiments:

- Additive noise: For each sample in the training set, Gaussian noise ($\mu = 0$, $\sigma = 1$) is added.
- Random pitch shift: Each traing sample has its pitch shifted randomly between $[-\frac{1}{4}, \frac{1}{4}]$ octaves, according to a uniform distribution.
- Time stretching: Samples were randomly either sped up or slowed down. The stretching coefficients were chosen uniformly in the interval $[0.5, 2]$.
- Random volume change: The volume of the recording is changed by a factor chose from the interval $[-10\text{ dB}, +10\text{ dB}]$

uniformly. The volume changing curves (constant, fade in, sine, cosine) were selected randomly for each sample.

Additionally, feature-based methods were used too. These were implemented in the data generator:

- Mixup: Proposed in [5], mixup has been commonly used in previous DCASE challenge submissions too [6–8]. We used alpha value of 0.4 as mixup parameter. In our implementation, mixup is performed by generating two data batches and randomly mixing both the features and the labels from those batches.
- Spectrum augmentation: Introduced in [9], spectrum augmentation is a technique where parts of input features are masked, forcing the network to learn more general details of the samples. In our experiments, we masked 10% of the input features along the time and frequency axis.
- Random temporal shuffle: Used previously in [10], random temporal shuffle divides the input feature map into 2 parts along the time axis and randomly shuffles them.

4. TEST-TIME AUGMENTATION

Test-time augmentation (TTA) is commonly used in image classification in order to increase the accuracy of a model predictions [11]. Contrary to data augmentations during training, TTA is applied during inference.

The main idea of TTA is that by making several randomly augmented copies of the input sample, then averaging the outputs for the augmented samples, more accurate predictions can be made without changing the model.

In our submissions, we used all the time-domain augmentations described in Section 3, as well as random temporal shuffle. The augmentations were carried out 10 times per sample. Also, in order to scale the output predictions to $[0, 1]$, softmax was applied after averaging.

5. ARCHITECTURES

5.1. Small MobileNetV2 model

In our experiments, we used a model based on MobileNetV2 architecture [12], but compared to the original, ours has a reduced number of layers. The block structure can be found in Table 1.

The main goal of MobileNetV2 is to give a compact model that can fit the limitations of mobile devices, yet have considerable accuracy on classification tasks. The model introduced in [12] is still too big for submission for DCASE 2021 Task 1A, therefore we reduced the number of layers according to Table 1. Another difference is that we used Leaky ReLU activation with $\alpha = 0.01$.

In order to fit the size limit, we quantized the parameters of the model to `float16`, thus our TensorFlow Lite model with $47,939$ parameters weights under 93.6 kB $(95,878$ bytes).

5.2. CP-ResNet model

We used the receptive-field regularized ResNet model introduced in [13] and used in previous DCASE challenges [7, 8]. We used $\rho = 4$ receptive field regularization and for the small model size, we reduced the residual block channel sizes to 16. On top of the CP-ResNet backbone, the same top layer is added as to the MobileNetV2 (see Table 1 layers 7–13).

This model has $58,266$ parameters, which were quantized to `float16` in order to fit the size limit of the task. The quantized TensorFlow Lite model weights just under 113.8 kB $(116,532$ bytes).

6. EXPERIMENTS

A key element in our submission is the usage of metric learning. We used Additive Angular Margin Loss (ArcFace) [14] in our training workflow. The main idea behind ArcFace is to force the network to learn a metric, which maps the input samples to the surface of a hypersphere. Class separability is ensured by adding a margin around each of the class manifolds.

Training using ArcFace is accomplished by adding a batch normalization and a fully connected layer on top of the network body. The number of perceptrons in the fully connected layer corresponds to the dimension of the hypersphere where the loss is calculated, in other words, it's called embedding space. On top of these, the ArcFace header should be added and trained using categorical cross-entropy. After training, the ArcFace header should be removed and replaced with a fully connected layer, only which should be trained.

ArcFace itself has 2 hyperparameters: scaling factor (s) and angular margin (m). Additionally, we found that the dimension of the embedding space has an impact on numerical performance: preliminary experiments showed that increasing the embedding dimension over 64 results in no significant additional gain in accuracy.

We found that ArcFace is sensitive to the initial values of the network, and is prone to instability if the model is not close enough to the optimum. In order to avoid this instability, we used the following training workflow:

1. First, we pre-trained our model using focal loss [15] as the loss function, and using all augmentations described in section 3. Learning rates were set using a cosine-decay-restart scheduler between 10^{-2} and $10^{-}3$. Focal loss parameters were chosen as $\alpha = 1$ and $\gamma = 2$. This training step was run for 200 epochs.
2. Secondly, we cut the last 3 layers off the top of our model, froze the weights, then added the ArcFace header for training. In this stage, we trained the ArcFace header for 10 epochs, using the full training split of the dataset, without any sort of augmentation.

#	Block	Configuration
1	Input	$128 \times 431 \times 3$
2	Convolution	(3×3), with (2×2) strides
3	Inverted residual block	Channel expansion to 25
4	Inverted residual block	Channel expansion to 32
5	Inverted residual block	Channel expansion to 38
6	Convolution	(1×1), Ch. expansion to 56
7	AvgPool2D	(2×2) pooling
8	BatchNormalization	
9	Dropout	10%
10	Dense	64 output taps
11	BatchNormalization	
12	Dense	10 output taps
13	Activation	SoftMax

Table 1: Block structure of our reduced MobileNetV2 model.

3. Thirdly, we unfroze all the weights of the model and trained for 200 epochs.

4. Lastly, we removed the ArcFace header, froze the model weights and re-added the last 3 layers. In order to train them, we balanced the dataset and used all of the augmentations. In this training stage, we used exponential decay learning rate scheduler, with initial LR of 10^{-3} and decay rate of 3.33×10^{-5} for 30 epochs.

In our experiments, we used ArcFace parameters $s = 30$ and $m = 0.5$. We chose batch size as 24 and used SGD optimizer without momentum. We saved the model at the end of each epoch and chose the one with the highest accuracy on the test split to work with on subsequent training phases.

7. SUBMISSION SUMMARY

7.1. Submission 1: MobileNetV2 with focal loss (R_MNv2_fl)

We trained the reduced MobileNetV2 model using focal loss, subsampling-based balancing, according to the first step of the workflow described in Section 6. We found that in this case, TTA brings no significant increase in accuracy, but increases the log-loss of the model, therefore TTA was not utilized in the evaluation phase.

7.2. Submission 2: MobileNetV2 with ArcFace (R_MNv2_af)

In this submission, we trained the reduced MobileNetV2 model using ArcFace, all according to the workflow described in Section 6. Evaluation results were calculated using TTA.

7.3. Submission 3: CP-ResNet with focal loss (CPRes_fl)

We used the CP-ResNet model and – similarly to Submission 1 – we trained it using focal loss, subsampling-based balancing, all according to the first step described in Section 6. During evaluation, test-time augmentations were not used.

7.4. Submission 4: CP-ResNet with ArcFace (CPRes_af)

We refined the model of Submission 3 using ArcFace, with margin chosen as $m = 0.35$. In the evaluation phase, we utilized TTA for increased accuracy.

8. RESULTS AND DISCUSSION

The accuracies of the submitted systems can be seen in Table 2. For reference, the baseline system provided by the challenge organizers is presented too. Overall, compared to the baseline, all of our submitted models have higher accuracy on the test split of the development dataset. Although, the accuracy of some classes (metro and shopping mall) are lower on our models.

The confusion matrix of the ArcFace-trained MobileNetV2 (R_MNv2_af) model can be found in Table 3. For most of the scenes, there are specific classes which are confused in a higher percentage than others. For example, busses, trams and metros are often confused. Similarly, airports and shopping malls are also often misclassified to each other. Listening to the recordings of such scenes, we realized that classification is not trivial for human listeners either.

The device-wise classification accuracy of the same model can be found in Table 4. Generally, it can be said that accuracies on unseen devices are lower than on devices used for training. Additionally, in most cases, our model performs better at samples from real recording devices, compared to simulated ones.

It should be mentioned, that test-time augmentation increases both the accuracy and the loss too. For example, evaluating our ArcFace-trained MobileNetV2 (R_MNv2_af) model without TTA, the loss is 1.258. In contrast, using TTA on the same system results in a loss value of 2.021.

Despite the small model size, ArcFace still manages to overfit the training dataset (see Figure 1), which suggest that there is still room for improvement for this model structure.

	Baseline system	R_MNv2_fl	R_MNv2_af	CPRes_fl	CPRes_af
`airport`	40.5	54.1	53.0	56.4	52.4
`bus`	47.1	64.6	63.3	69.4	76.4
`metro`	51.9	48.8	52.5	27.3	51.5
`metro_station`	28.3	47.5	46.8	51.2	44.4
`park`	69.0	70.4	62.3	72.1	61.3
`public_square`	25.3	39.7	29.3	36.4	42.8
`shopping_mall`	61.3	50.8	49.8	60.6	50.5
`street_pedestrian`	38.7	41.1	42.1	42.1	30.0
`street_traffic`	62.0	79.1	84.8	78.5	81.8
`tram`	47.7	57.1	59.1	51.4	56.1
Average accuracy	47.7	55.3	54.3	54.5	54.7

Table 2: Scene classification accuracy comparison between systems. The values are percentages.

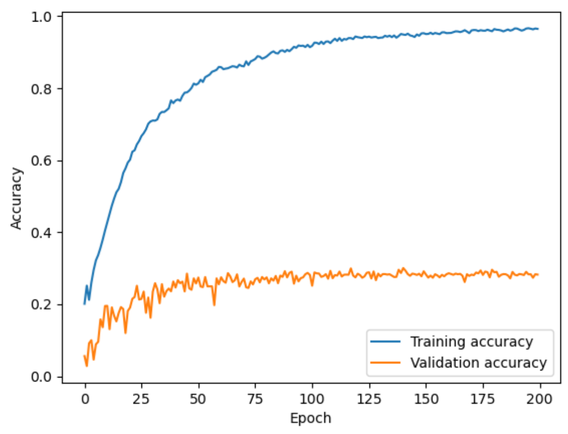

Figure 1: Training MobileNetV2 with ArcFace (R_MNv2_af): significantly overfitting the training data. Note that because of the additional margin, the accuracies here do not reflect the final accuracy of the network.

	airport	bus	metro	metro_station	park	public_square	shopping_mall	street_pedestrian	street_traffic	tram
airport	**53.0**	0.7	2.4	9.5	0.0	3.7	18.9	9.8	0.0	2.0
bus	0.0	**63.3**	9.1	1.0	0.7	0.7	0.0	1.0	1.0	23.2
metro	0.1	8.8	**52.5**	10.4	0.0	2.4	0.7	4.4	0.0	19.9
metro_station	10.1	3.7	11.8	**46.8**	0.7	1.3	10.8	7.4	3.4	4.0
park	0.7	4.7	2.0	4.0	**62.3**	6.4	1.0	2.0	15.2	1.7
public_square	7.7	2.7	2.7	11.4	5.7	**29.3**	3.7	9.1	25.9	1.7
shopping_mall	14.5	0.3	2.4	12.1	0.0	0.7	**49.8**	18.5	1.0	0.7
street_pedestrian	7.1	3.4	2.4	4.0	0.3	17.2	11.1	**42.1**	11.1	1.3
street_traffic	0.0	0.0	0.0	2.0	3.0	5.4	1.0	2.7	**84.8**	1.0
tram	1.4	13.2	11.1	7.1	3.4	2.0	0.7	1.0	1.0	**59.1**

Table 3: Confusion matrix of ArcFace-trained MobileNetV2 (R_MNv2_af) model on the test split of the development set. The values are percentages and are normed.

	airport	bus	metro	metro_station	park	public_square	shopping_mall	street_pedestrian	street_traffic	tram
A	78.8	87.9	75.7	54.5	75.8	60.6	60.6	63.6	90.9	66.7
B	71.9	54.5	51.5	33.3	57.6	24.2	45.5	51.5	87.9	69.7
C	72.7	81.8	66.7	39.4	81.8	51.5	33.3	54.5	87.9	46.9
S1	48.5	60.8	45.5	54.5	66.7	27.3	48.5	45.5	93.9	57.6
S2	48.5	75.8	54.5	42.4	54.5	21.2	45.5	42.4	75.8	51.5
S3	57.6	60.6	60.6	60.6	66.7	15.2	48.5	36.2	78.8	78.8
S4	21.2	51.5	48.5	48.5	54.5	30.3	54.5	33.3	93.9	54.5
S5	42.4	48.5	51.5	39.4	72.7	24.2	66.7	30.3	84.8	48.5
S6	36.4	48.5	18.2	48.5	30.3	9.1	45.5	21.2	69.7	57.6

Table 4: Device-wise scene classification accuracies of our ArcFace-trained MobileNetV2 (R_MNv2_af) model. The values are percentages.

9. CONCLUSION

In this technical report, we presented our submissions for DCASE 2021 challenge Task 1A. We designed and trained 10-class acoustic scene classifier models, which can achieve around 54–55% accuracy on the development dataset.

In our experiments, we trained our models using ArcFace metric learning method and evaluated using test-time augmentations and averaging. The results suggest that if overfitting can be avoided, higher accuracies could be obtained when using metric learning methods.

10. REFERENCES

[1] I. Martín-Morató, T. Heittola, A. Mesaros, and T. Virtanen, "Low-complexity acoustic scene classification for multi-device audio: analysis of DCASE 2021 Challenge systems," 2021.

[2] T. Heittola, A. Mesaros, and T. Virtanen, "Acoustic scene classification in DCASE 2020 Challenge: generalization across devices and low complexity solutions," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop (DCASE2020)*, 2020, submitted. [Online]. Available: https://arxiv.org/abs/2005.14623

[3] "TensorFlow," Accessed: 14-June-2021. [Online]. Available: https://www.tensorflow.org

[4] "librosa," Accessed: 14-June-2021. [Online]. Available: https://librosa.org

[5] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[6] H. Hu, C.-H. H. Yang, X. Xia, X. Bai, X. Tang, Y. Wang, S. Niu, L. Chai, J. Li, H. Zhu, F. Bao, Y. Zhao, S. M. Siniscalchi, Y. Wang, J. Du, and C.-H. Lee, "Device-Robust Acoustic Scene Classification Based on Two-Stage Categorization and Data Augmentation," DCASE2020 Challenge, Tech. Rep., June 2020.

[7] K. Koutini, F. Henkel, H. Eghbal-zadeh, and G. Widmer, "CP-JKU submissions to DCASE'20: Low-complexity cross-device acoustic scene classification with rf-regularized CNNs," DCASE2020 Challenge, Tech. Rep, Tech. Rep., 2020.

[8] K. Koutini, H. Eghbal-zadeh, and G. Widmer, "Acoustic Scene Classification and Audio Tagging with Receptive-Field-Regularized CNNs," DCASE2019 Challenge, Tech. Rep., June 2019.

[9] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[10] T. Inoue, P. Vinayavekhin, S. Wang, D. Wood, N. Greco, and R. Tachibana, "Domestic Activities Classification Based on CNN Using Shuffling and Mixing Data Augmentation," DCASE2018 Challenge, Tech. Rep., September 2018.

[11] D. Shanmugam, D. Blalock, G. Balakrishnan, and J. Guttag, "When and Why Test-Time Augmentation Works," *arXiv preprint arXiv:2011.11156*, 2020.

[12] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[13] K. Koutini, H. Eghbal-Zadeh, M. Dorfer, and G. Widmer, "The receptive field as a regularizer in deep convolutional neural networks for acoustic scene classification," in *2019 27th European signal processing conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[14] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4690–4699.

[15] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.