INTEGRATING THE DATA AUGMENTATION SCHEME WITH VARIOUS CLASSIFIERS FOR ACOUSTIC SCENE MODELING

Technical Report

Hangting Chen[1,2], *Zuozhen Liu*[1,2], *Zongming Liu*[1,2], *Pengyuan Zhang*[1,2*], *Yonghong Yan*[1,2,3]

[1]Key Laboratory of Speech Acoustics & Content Understanding, Institute of Acoustics, CAS, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]Xinjiang Laboratory of Minority Speech and Language Information Processing,
Xinjiang Technical Institute of Physics and Chemistry, CAS, China

ABSTRACT

This technical report describes the IOA team's submission for TASK1A of DCASE2019 challenge. Our acoustic scene classification (ASC) system adopts a data augmentation scheme employing generative adversary networks. Two major classifiers, 1D deep convolutional neural network integrated with scalogram features and 2D fully convolutional neural network integrated with Mel filter bank features, are deployed in the scheme. Other approaches, such as adversary city adaptation, temporal module based on discrete cosine transform and hybrid architectures, have been developed for further fusion. The results of our experiments indicates that the final fusion systems A-D could achieve an accuracy higher than 85% on the officially provided fold 1 evaluation dataset.

Index Terms— Acoustic scene classification, Convolutional neural network, Generative adversary network, Wavelet, Mel filter bank

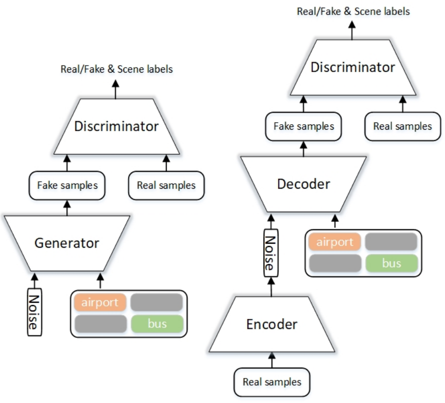
1. INTRODUCTION

Acoustic scene classification (ASC) aims to classify sounds into one of predefined classes [1]. Detection and Classification of Acoustic Scenes and Events (DCASE) challenges organized by IEEE Audio and Signal Processing (AASP) Technical Committee are one of the biggest competitions for ASC task. The large-scale dataset provided by DCASE2019 presents a difficult challenge for the system's fitting ability and generalization.

The report describes the details of IOA team's submission for TASK1A of DCASE2019. More concretely, data augmentation schemes based on generative neural networks (GAN) as well as two major classifiers improve the system's performance. We use two types of features, Mel filter bank feature (FBank) and scalogram extracted by wavelets, and two types of neural networks, 2D fully convolutional neural networks (FCNN) and 1D deep convolutional neural networks (DCNN). Other techniques, such as adversary domain adaptation, temporal module based on discrete cosine transform (DCT), hybrid neural network architectures, are developed for model ensemble. Under the official fold 1 evaluation setup, the final fusion systems could achieve above 85% accuracy in the evaluation set.

The remainder of this report is organized as follows. Section 2 describes the scheme of data augmentation. Section 3 details the features and architectures of classifiers. Section 4 presents our two methods for fusion. Section 5 shows the details of experiments. Section 6 covers the results of classifiers and fusion systems and makes some discussion. Section 7 concludes our work.

2. THE DATA AUGMENTATION SCHEME



(a). ACGAN (b). CVAE/ACGAN

Figure 1: (a) ACGAN and (b) CVAE/ACGAN architecture for data augmentation.

Though most ASC systems can accurately classify training samples, they suffer from inferring test records, especially those from unseen cities [2][3]. To improve generalization, additional samples are generated and added into the database. Inspired by the rapid development of generative models in deep learning, auxiliary classifier GAN (ACGAN) [4] are utilized to generate fake samples (Figure 1(a)). The generator learns to create acoustic feature maps which look real with scene labels as an additional input condition. On the other hand, the discriminator learns to distinguish real features from fakes as well as scene labels. Thus the generator/discriminator aims to maximum/minimize the binary real/fake loss

$$L_{real/fake} = \sum_i (log(Dis(x_i)) + log(1 - Dis(Gen(y_i, z)))), \tag{1}$$

and to minimize the scene classification loss,

$$L_{scene} = \sum_i \sum_{a \in A} \mathbb{I}_{[a=y_i]}(log(Dis(x_i))+log(Dis(Gen(y_i, z)))), \tag{2}$$

where x_i,y_i,z represents the networks' input, target and Gaussian noise, A is a collection of scene classes, $\mathbb{I}$ is the indicator function. The loss of ACGAN is defined as,

[*] Pengyuan Zhang is the corresponding author.

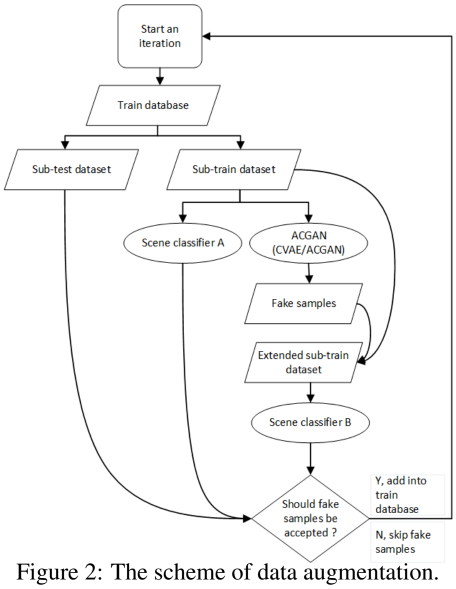


Figure 2: The scheme of data augmentation.

$$L_{ACGAN} = L_{real/fake} + \gamma L_{scene}, \tag{3}$$

where γ controls the ratio between real/fake loss and scene classification loss.

The noise to generate fake features is usually assumed following Gaussian distribution, which actually makes little sense. To obtain samples as real as possible, a CVAE/ACGAN architecture (Figure 1(b)) is deployed as an alternative to ACGAN. It uses encoder to encode the real samples into noise restricted by a Kullback-Leibler (KL) loss,

$$L_{KL} = \sum_i D_{KL}(Enc(x_i)||\mathcal{N}(0, I)), \tag{4}$$

with respect to encoder with x_i being the training sample. Besides, a reconstruction loss enables the reconstructed ones similar in contrast with the original input. In our framework, a mean-square loss of high-level bottleneck feature in the lth layer of the discriminator serves as reconstruction loss [5],

$$L_{reco} = \sum_i (Dis_l(x_i) - Dis_l(\tilde{x}_i))^2, \tag{5}$$

where $\tilde{x}_i$ is the reconstruction of x_i. Therefore, the CVAE/ACGAN loss is defined as,

$$\begin{aligned} L_{CVAE/ACGAN} =& L_{real/fake} + \gamma_1 L_{scene} \\ &+ \gamma_2 L_{KL} + \gamma_3 L_{reco}, \end{aligned} \tag{6}$$

where γ_1,γ_2,γ_3 are the coefficients to balance various losses.

The complete framework is plotted in Figure 2, denoted as ACGAN or CVAE/ACGAN data augmentation scheme. In each iteration, the train database is firstly split into non-overlapped sub-train and sub-test set. Then a base classifier A is trained and tested on the sub-datasets. Also a generative model is trained on the sub-train set and then sampled when its output is stabilized. The fake samples are examined before added into the whole dataset. These generated candidates are mixed into sub-train set and another scene classifier B is trained. If its performance is improved on the sub-test dataset, these fake candidates will be accepted and added into the whole database.

Compared with [6], the fakes from different scenes can be sampled directly from the ACGAN (or CVAE/ACGAN) with scene labels as the condition. No need for training individual GANs for each scene class. Moreover, the discriminator with an auxiliary classifier ensures the generated samples not only look real but also belong to the target scene labels.

3. CLASSIFICATION SYSTEMS

3.1. FBank-FCNN Classifier

The FBank-FCNN network architecture is shown in Table 1, similar to the one proposed in [7]. It is a VGG [8] style Network with 10 repeatedly stacked convolution layers containing small convolutional kernels. The common techniques in deep learning, such as batch normalization, dropout and Rectified Linear Units (ReLU), are used following the convolutional operations. The final classification part is designed as an 1×1 convolutional layer by decreasing the amount of channels to 10 followed by a global average pooling layer over 10 feature maps, and finally a 10-way SoftMax to the segment-level prediction.

Table 1: The FCNN Classifier. The input feature map is of size frames(L) × channels(c) × filters(n). The notation ”5 × 5 Conv(pad=2,stride=2)×$14c$-BN-ReLU” denotes a convolutional kernel with $14c$ output channels and a size of 5 × 5,followed by batch normalization and ReLU activation.

Layer Name	Settings
Input	Fbank $L \times c \times n$
Conv1	5 × 5 Conv(pad=2,stride=2)×$14c$-BN-ReLU 3 × 3 Conv(pad=1,stride=1)×$14c$-BN-ReLU 2 × 2 MaxPooling
Conv2	3 × 3 Conv(pad=1,stride=1)×$28c$-BN-ReLU 3 × 3 Conv(pad=1,stride=1)×$28c$-BN-ReLU 2 × 2 MaxPooling
Conv3	3 × 3 Conv(pad=1,stride=1)×$56c$-BN-ReLU Dropout($p = 0.3$) 3 × 3 Conv(pad=1,stride=1)×$56c$-BN-ReLU Dropout($p = 0.3$) 3 × 3 Conv(pad=1,stride=1)×$56c$-BN-ReLU Dropout($p = 0.3$) 3 × 3 Conv(pad=1,stride=1)×$56c$-BN-ReLU 2 × 2 MaxPooling
Conv4	3 × 3 Conv(pad=0,stride=1)×$128c$-BN-ReLU Dropout($p = 0.5$) 3 × 3 Conv(pad=0,stride=1)×$128c$-BN-ReLU Dropout($p = 0.5$)
Pooling	1 × 1 Conv(pad=0,stride=1)×10-BN-ReLU GlobalAveragePooling
Output	10-way SoftMax

3.2. Scalogram-DCNN Classifier

The Scalogram-DCNN classifier is based on [9]. The scalogram, which is used as an input of the DCNN classifier, is locally translation invariant and stable to time-warping deformation [10]. In this system, it is generated from wavelet filters operating on the spectrogram which is transformed from raw waveform. As shown in Table 2, the DCNN classifier consists of convolutional layers with small kernels and fully-connected (FC) layers.

3.3. Other Classifiers

3.3.1. DCT Temporal Module

The scalogram-DCNN classifier is trained and evaluated in a frame-wise way. Due to the long term characteristics of wavelets, recurrent neural network can not achieve high classification accuracy. The temporal module based on DCT is deployed after the final affine transform as described in [9]. Different from the DCT filter in image processing, an attention weight filters the DCT spectrum by strengthening and weakening target feature map bins.

3.3.2. Adversary City Adaptation

To generalize the classifier for unseen cities, an adversary training branch, composed of a gradient reverse layer and a 2-layer feed-forward classifier, is connected following the convolutional layers in the scalogram-DCNN system. The branch classifies the record into the target city while a gradient reverse layer [11] makes the output of convolutional layers similar for the same scene class over various city domains.

Table 2: The DCNN Classifier. The input feature map is of size frames(L) $\times$ channels(c) $\times$ filters(n). The notation ”$c \times 3$ Conv(pad=0,stride=1)-$2c$-BN-ReLU” denotes a convolutional kernel with c input channels, $2c$ output channels and a size of 3, followed by batch normalization and ReLU activation.

Layer Name	Settings
Input	Scalogram $L \times c \times n$
Conv1	$c \times 3$ Conv(pad=0,stride=1)$\times 2c$-BN-ReLU 2 Pooling(pad=1,stride=2)
Conv2	$2c \times 3$ Conv(pad=0,stride=1)$\times 4c$-BN-ReLU 2 Pooling(pad=0,stride=2)-Dropout
Conv3	$4c \times 3$ Conv(pad=0,stride=1)$\times 8c$-BN-ReLU 2 Pooling(pad=0,stride=2)
Conv4	$8c \times 3$ Conv(pad=0,stride=1)$\times 16c$-BN-ReLU 2 Pooling(pad=0,stride=2)-Dropout
	Concatenate and flatten input as well as Conv's output
FC1	Linear (1024 units)-BN-ReLU-Dropout
FC2	Linear (1024 units)-BN-ReLU-Dropout
FC3	Linear (1024 units)-BN-ReLU
Output	10-way SoftMax

3.3.3. Hybrid Network Architecture

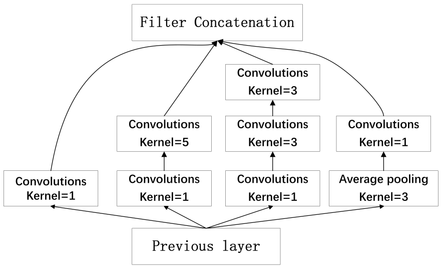


Figure 3: Inception module I network architecture.

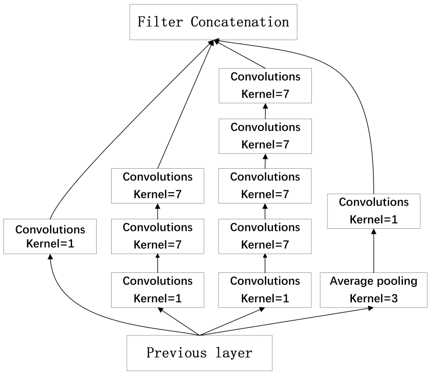


Figure 4: Inception module II network architecture.

Several hybrid network architectures are proposed as classifiers. In many machine learning tasks using deep learning, increasing the size of networks can achieve better classification results. However, it may lead to a large amount of parameters which gives rise to the risk of overfitting due to limited labeled data. A deep convolutional network architecture codenamed Inception increased the depth and width of the network while keeping the computational budget constant [12]. Then, improved versions of the Inception were proposed in [13][14]. In order to expand the base network architectures with little parameter growth, we use two Inception modules (Figure 3 and Figure 4) to replace the last 2 convolutional layers in the Scalogram-DCNN classifier. Unlike the work in the Inception networks, we try 1D and 2D CNN layers respectively in these Inception modules to maintain compatibility with the 1D CNN layers we use in the base DCNN architecture.

It is noticed that our DCNN network architecture with 1D convolutional layers may not make good use of temporal information. So recurrent layers like long short-term memory (LSTM) and gated recurrent unit (GRU) are added as a parallel channel for FC layers. Specifically, these layers use the output of the final convolutional layer as input. And their output as well as that of FC layers are combined as input to the last layers.

After combining the Inception modules and recurrent layers into the base DCNN network, we get several hybrid network architectures, named IncepLSTM and IncepGRU. The IncepLSTM classifier uses 2 layers of Inception module I instead of simple convolution and 2 layers of LSTM added as the parallel channel. The IncepGRUV1 classifier use 2 layers of Inception module II and 2 layers of GRU as the parallel channel. The IncepGRUV2 reduces the number of Inception module I to 1 followed with 1 layer of Inception module II. The IncepGRUV3 subsystem changes the convolutional layers of the Inception modules in V1 to 2D convolution. They are all expected to have some complementarity with the base classifiers.

4. ENSEMBLE METHODOLOGY

Different classifiers may lead to divisions among some controversial samples. Model ensemble can stabilize and generalize our final results. In the practice, voting serves as a simple and effective method compared with support vector machine, regression, etc. Two strategies, average and weighted voting, are adopted. The latter's weight is trained on the fold 1 evaluation set.

5. EXPERIMENTS

We used the officially provided fold 1 procedure to evaluate our systems' performance. Then the systems were retrained on the whole development data for submission. The train set was firstly split into the train and validate set. The classifiers were trained on the train set in maximum 200 epochs. The validation set determined the early stopping of the training, i.e., the training would be stopped if its loss failed to decrease in continuous 5 epochs. We used Adam optimizer and set β_1 and β_2 to 0.9 and 0.999. The initial learning rate was 10^{-3} and was decreased according to the loss on the validate set. To relieve the influence of model's initialization, each system was trained in 3 different initial seeds. External data was not used in all experiments.

5.1. Data augmentation

In each iteration of ACGAN (or CVAE/ACGAN) scheme, the train set was divided into sub-train and sub-test set of approximately equal size according to their recording cities. A base classifier was trained before and after the fake samples were added. Only if a performance improvement was observed, the fake samples were accepted and added into the whole database. In each iteration, we trained the ACGAN (or CVAE/ACGAN) for nearly 50 epochs and sampled sets of spectrum from models in different training epochs.

5.2. FBank-FCNN experiments

We built two systems of different types of input features under FBank-FCNN architecture, one with left and right channel features as the input, another with difference and sum channel input. For both features, STFT was applied on the signal every $20ms$ over $40ms$ hamming windows. The total number of filters was 128. The Fbank of a 10-second stereo audio was of the dimension $500 \times 2 \times 128$ before and $500 \times 6 \times 768$ after adding delta and delta-delta coefficients. For data augmentation, the generator and the discriminator were simplified versions of the classifier in Section 3.1.

5.3. Scalogram-DCNN experiments

To extract the scalogram, STFT was applied on the raw signal every $185ms$ over $555ms$ windows. The total number of wavelet filters was set to 290, distributed uniformly at low frequency and logarithm at high frequency as described in [15]. The scalogram of a 10-second stereo record was in a dimension of $58 \times 2 \times 290$. We followed the detailed settings of wavelets in [9]. The segment-wise prediction was obtained by accumulating the frame-wise output from the scalogram-DCNN classifier. Also the generator and discriminator adopted a simplified version of the classifier in Section 3.2.

6. RESULTS AND DISCUSSION

Table 3: Results of experiments of different data augmentation frameworks on the fold 1 evaluation set, where the best performance is in bold.

Feature type	Channels	Data augmentation scheme	Accuracy(%)
Fbank	Left-Right	w/o	76.92
Fbank	Left-Right	ACGAN	**77.56**
FBank	Ave-Diff	w/o	79.95
FBank	Ave-Diff	ACGAN	**80.10**
Scalogram	Left-Dight	w/o	77.03
Scalogram	Left-Dight	ACGAN	**80.98**
Scalogram	Ave-Diff	w/o	82.28
Scalogram	Ave-Diff	ACGAN	84.06
Scalogram	Ave-Diff	CVAE/ACGAN	**84.28**

In this section, the complete results of the fold 1 evaluation setup on different schemes and classifiers are reported. The feature extracted from left-right and ave-diff (average-difference) channels was evaluated under FBank-FCNN and scalogram-DCNN classifiers with and without data augmentation. As listed in Table 3, the features extracted from ave-diff channels could outperform that from left-right channels, approximately 3%-5%. In addition, the GAN scheme can improve all classifiers performance from 0.5% to 4%. The CVAE/ACGAN scheme could further give rise to higher accuracy in the scalogram-DCNN.

Two strategies were deployed in our following experiments. First, the ACGAN was adopted as a main scheme instead of CVAE/ACGAN, whose training was relatively slow. Moreover, ACGAN was already pretty distinct and when integrated with other techniques, it seemed more preferable to CVAE/ACGAN. Second, the Scalogram-DCNN classifiers trained only on the ave-diff features were served as a main force of system ensemble and FBank-FCNN classifiers trained both on the ave-diff and left-right features as supplementary. The results are listed in Table 4, where we denote the name of each system as ”{feature type}-{feature channel}-{data augmentation scheme}-{classifiers}”. For example, ”scalogram-avediff-ACGAN-city_adversary” represents the system trained under ACGAN scheme using ave-diff scalogram and the adversary city adaptation classifier.

Table 4: Results of experiments of various systems on the fold 1 evaluation set, where the top 3 classifiers are in bold.

Classifier ID	Classifier Name	Accuracy(%)
1	Fbank-leftright-ACGAN-FCNN	77.56
2	Fbank-avediff-ACGAN-FCNN	80.10
3	Scalogram-avediff-ACGAN-DCNN	84.06
4	Scalogram-avediff-ACGAN-DCNN_DCT	83.58
5	Scalogram-avediff-ACGAN-IncepLSTM	83.80
6	Scalogram-avediff-ACGAN-IncepGRU	83.87
7	Scalogram-avediff-ACGAN-city_adversary	**84.16**
8	Scalogram-avediff-ACGAN-city_adversary_DCT	**84.23**
9	Scalogram-avediff-CVAE/ACGAN-DCNN	**84.28**
10	Scalogram-avediff-CVAE/ACGAN-city_adversary	82.94

Table 5: Results of fusion systems on the fold 1 evaluation set. System 2 and 3 used different weight.

System ID	Classifiers' IDs	Voting methods	Accuracy(%)
A	1,2,3,4,5,6,7	Average	85.07
B	1,2,3,4,7,8	Weighted	85.11
C	1,2,3,4,7,8	Weighted	85.11
D	1,2,3,4,5,6,7,9	Average	**85.28**

After data augmentation, the system built on wavelet features could always outperform the ones using FBank (1,2 and 3-10). The adversary city adaptation (3 and 7) and CVAE/ACGAN scheme (3 and 9) could lead to improvement but combing both failed to give better results (7,9,10). Additionally, the hybrid networks did not outperform the scalogram-avediff-ACGAN-DCNN ones (5,6 and 3). The DCT temporal module slightly promote the accuracy (7 and 8) but may harm the systems in some cases (3 and 4). The top 3 classifiers were scalogram-avediff-CVAE/ACGAN-DCNN, scalogram-avediff-ACGAN-city_adversary_DCT and scalogram-avediff-ACGAN-city_adversary, which all employed wavelet filters, average-difference channels' features and data augmentation scheme.

In fusion systems, FBank and scalogram features could be relatively complemented under a proper combination strategy. The detailed voting systems for the final submission are listed in Table 5, two using the weighted voting and two using the average voting.

7. CONCLUSION

This report describes our submissions for DCASE2019 Task1A. In the fold 1 evaluation setup, the data augmentation scheme integrated with convolutional neural networks and other training architectures achieved accuracies above 77% and 83% for FBank and scalogram. After voting fusion, the final systems could achieve accuracies above 85%.

8. REFERENCES

[1] A. Mesaros, T. Heittola, and T. Virtanen, “Tut database for acoustic scene classification and sound event detection,” pp. 1128–1132, 2016.

[2] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, “DCASE2017 challenge setup: Tasks, datasets and baseline system,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, November 2017, pp. 85–92.

[3] 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.

[4] A. Odena, C. Olah, and J. Shlens, “Conditional image synthesis with auxiliary classifier gans,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70.* JMLR. org, 2017, pp. 2642–2651.

[5] A. B. L. Larsen, S. K. Sonderby, H. Larochelle, and O. Winther, “Autoencoding beyond pixels using a learned similarity metric,” *international conference on machine learning*, pp. 1558–1566, 2016.

[6] S. Mun, S. Park, D. Han, and H. Ko, “Generative adversarial network based acoustic scene training set augmentation and selection using SVM hyper-plane,” DCASE2017 Challenge, Tech. Rep., September 2017.

[7] M. Dorfer, B. Lehner, H. Eghbal-zadeh, H. Christop, P. Fabian, and W. Gerhard, “Acoustic scene classification with fully convolutional neural networks and I-vectors,” DCASE2018 Challenge, Tech. Rep., September 2018.

[8] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *international conference on learning representations*, 2015.

[9] H. Chen, P. Zhang, and Y. Yan, “An audio scene classification framework with embedded filters and a dct-based temporal module,” in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2019, pp. 835–839.

[10] H. Chen, P. Zhang, H. Bai, Q. Yuan, X. Bao, and Y. Yan, “Deep convolutional neural network with scalogram for audio scene modeling.” in *Interspeech*, 2018, pp. 3304–3308.

[11] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, “Adversarial discriminative domain adaptation,” *computer vision and pattern recognition*, pp. 2962–2971, 2017.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[14] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[15] J. Andén and S. Mallat, “Deep scattering spectrum,” *CoRR*, vol. abs/1304.6763, 2013. [Online]. Available: http://arxiv.org/abs/1304.6763