Environmental Sound Classification Using Convolutional Neural Network Based Transfer Learning

Jingyang Zhou[†], Jianrui Lu[‡], Ruisong Wang[†], Ruofei Ma[†*], and Zhiliang Qin[§]
[†]Department of Communication Engineering, Harbin Institute of Technology, China
Email: zjyfss@163.com, mathwrs@163.com, maruofei@hit.edu.cn
[‡]School of Engineering, Guangzhou College of Technology and Business, China. Email: ljrice@126.com
[§]Waihai Beiyang Electric Group Co. Ltd., China. Email: qinzhiliang@beiyang.com

***Abstract*—Environmental sound classification (ESC) plays a key role in various daily applications but faces challenges such as limited datasets and the complex temporal nature of sounds. Traditional ESC systems, which often rely on SVM or KNN classifiers, struggle with accuracy due to hardware limitations. Recent advancements like convolutional neural networks (CNN) and transfer learning provide effective solutions. This paper presents a CNN-based approach with transfer learning for ESC, using three features: mel spectrogram (MEL), mel-frequency cepstral coefficient (MFCC), and scalogram. These features are converted into RGB images and processed with Xception as the pre-trained model. Evaluations on the ESC-10 and ESC-50 datasets show accuracies of 93.3% and 84.5%, respectively, highlighting the effectiveness of our approach in improving classification accuracy and offering a reliable solution for sound recognition.**

***Index Terms*—Environmental Sound Classification (ESC), Convolutional Neural Network (CNN), Transfer Learning, mel spectrogram(MEL), mel-frequency cepstral coefficient(MFCC), scalogram**

I. Introduction

During recent several years, more and more attention is paid on environmental sound classification (ESC). There are lots of useful information contained in environmental sounds, which can be used to understand current situations that are happening at a particular time and place. ESC has been widely applied in many fields like machine hearing in distress circumstance detection for the elderly people at home, sound event recognition for audio surveillance, person tracking, smart city, animal sound classifying, animal activity detecting and so on. In the past, machine learning has been widely adopted in ESC task. For example, support vector machine(SVM) [1], a generalized linear classifier, is used to discriminate environmental sounds. Besides, there are a lot of other machine learning methods used in ESC task, such as K-nearest neighbors(KNN) [2], [3], hidden Markov models (HMM) [4] and Gaussian mixture models(GMM) [5]. Those traditional machine learning algorithms are easy to use and have low requirements for computing capacity, so they can be widely adopted. However, the system using those traditional classification schemes have commonly a low classification accuracy. A major reason is that typical classifiers(e.g. SVM, KNN, HMM and GMM) lack the ability of extracting more comprehensive and accurate features of environmental sounds.

Convolutional neural networks (CNN) have gained significant attention in recent years for their high accuracy in various tasks. For example, SoundNet [6], proposed by Yusuf et al., is a deep CNN that achieves 92.2% accuracy on the ESC-10 dataset and 74.2% on ESC-50, outperforming traditional machine learning methods. Thus, CNN is a promising approach for ESC tasks.

The ESC task often faces the challenge of limited labeled datasets, making it difficult to extract useful information from sound clips. Obtaining fully labeled data is costly and impractical at scale. Consequently, researchers have explored more efficient methods, such as using weakly labeled data, combining image and acoustic features, and applying local binary patterns for temporal feature fusion.

To address the challenge of limited labeled data and improve classification accuracy, we propose an ESC system using CNN with transfer learning. Transfer learning allows a model to leverage knowledge from previous tasks, making it effective for training on small datasets. Typically, a pre-trained model is required for transfer learning. In this paper, we use Xception, an improved version of Inception, as the pre-trained model. Xception has shown superior performance on ImageNet and JFT datasets and is faster than Inception for recognition tasks. Additionally, Keras provides an easy-to-use Xception API, making it a convenient choice for our system.

Our ESC system consists of four main steps: signal pre-processing, feature extraction, RGB image formation, and classification model training. We use the ESC-50 [8] and ESC-10 [8] datasets, which are divided into 5 folds for cross-validation. Signal pre-processing splits the datasets into 5 folds and segments long input signals into frames for feature extraction. Feature extraction reduces the size of the training data and re-represents the complex data in suitable formats. We extract three common acoustic features: mel spectrogram (MEL), mel-frequency cepstral coefficients (MFCC), and scalogram. These

This work is supported by Shandong Provincial Natural Science Foundation (ZR2023MF001, ZR2020MF141), the National Natural Science Foundation of China (61971156, 61801144). (*Corresponding authors: Ruofei Ma*)

features are then combined to represent the original signal as an RGB image. MEL is widely used for sound classification and is obtained by first applying Short-Time Fourier Transform (STFT) and then passing the spectrogram through mel-scale filter banks. MFCC captures the static characteristics of sound and is computed through six steps: pre-emphasis, framing, windowing, Fourier Transform (FT), passing through the mel filter bank, and applying Discrete Cosine Transform (DCT). To extract the scalogram feature, we apply continuous wavelet transform (CWT) to the input signal and resize it to 299 $\times$ 299. After obtaining MEL, MFCC, and scalogram, the features are fused into an RGB image of size 299 $\times$ 299. The details of the extraction and conversion processes are provided in a later section.

As mentioned, our CNN uses transfer learning with Xception as the pre-trained model. The RGB images derived from the features (MEL, MFCC, and scalogram) serve as input to Xception. The output of Xception is resized and fed into our CNN for training. We use the ESC-50 and ESC-10 datasets to train and evaluate the network. The achieved accuracy is - for the ESC-10 dataset and 84.5% for ESC-50. Compared to the Piczak method [10], our system performs better, with an improvement of - for ESC-10 and 20% for ESC-50. These results demonstrate that transferring knowledge from image recognition models and training deeper neural networks improve performance for the ESC task. The CNN architecture and experimental procedure are detailed in later sections.

II. Methodology

A. Datasets

In our experiment, we used two well-known environmental sound datasets: ESC-10 and ESC-50, provided by Karol J. Piczak [8]. One major challenge in developing effective environmental sound classification systems is the limited availability of datasets. Relying solely on these limited datasets makes it difficult to train a high-performance model without incorporating additional techniques. Both the ESC-10 and ESC-50 datasets have limitations. The ESC-50 dataset contains only 2000 sound recordings from 50 categories, which can be grouped into five major categories [8]:

- exterior/urban noises
- animal sounds
- interior/domestic sounds
- natural soundscapes and water sounds
- human (non-speech) sounds

Each major category includes 10 classes(40 sound clips per class). The ESC-10 dataset is a smaller dataset, which select 10 classes from the bigger one. It contains total 400 environmental sound recordings representing three general groups of sounds: sound continuing for only a short time, structured noise, sound events with strong harmonic content. These 10 classes are: sneezing, helicopter, clock ticking, dog barking, baby crying, rooster crowing, sea waves, rain and chainsaw. Beside, both ESC-50 and ESC-10 are pre-sorted in 5 folds for cross-validation. Hence, we train and evaluate our model with a 5-fold cross-validation regime.

B. Audio Representation

In 2017, Nanni et al. proposed a novel method for audio classification [7]. By combining visual and acoustic features, they found that feature fusion improved classification accuracy. This suggests that fusing acoustic and image features is an effective method to enhance classification system performance. In our paper, we use mel spectrogram (MEL) [9], mel-frequency cepstral coefficient (MFCC) [10], and scalogram [9] as input features. After extracting these features, we fuse them into an ensemble and represent the environmental sound clip as an RGB image, with the R channel corresponding to MEL, the G channel to scalogram, and the B channel to MFCC.For an ESC system, selecting an appropriate method to represent the audio signal is crucial. Using the right representation can significantly improve system accuracy [11].

MEL and MFCC are widely used in acoustic recognition systems and typically lead to high accuracy, which is why we select them.Since sound signals are one-dimensional time-domain signals, the frequency content is not easily discernible. To address this, Fourier Transform (FT), which converts the signal to the frequency domain, is commonly used. FT is defined as:

$$F(\omega) = \int_{-\infty}^{+\infty} f(t) e^{-j\omega t} dt \tag{1}$$

where $F(\omega)$ is the function of frequency domain and $f(t)$ is the origin signal.

As a method of signals analyzing, FT can be used to analyze the components of the signal and synthesize the signal. However, while FT reveals the frequency distribution, it loses time-domain information. To address this, Short-Time Fourier Transform (STFT), a classic time-frequency analysis method, is used. STFT applies Fourier Transform to short segments of the signal. The process is simple: divide the long sound signal into frames, perform Fourier Transform on each frame, and stack the results in a new dimension. This produces a two-dimensional signal called a spectrogram. The mathematical definition of STFT is:

$$X_i(\omega) = \sum_{n=-\infty}^{+\infty} x(n)\omega(n - iR)e^{-j\omega n} \tag{2}$$

where $X_i(\omega)$ is the STFT of one frame centered about time iR. $x(n)$ is the origin input signal at time n and $\omega(n)$ is a window function(e.g. Chebyshev) with length I. R is the hop size, in samples, between different frames.

By passing through mel-scale filter banks, the spectrogram is transformed into the MEL. The mel-scale is based on the non-linear way the human ear perceives sound. Humans are more sensitive to low-frequency sounds and less sensitive to high frequencies. Therefore, the mel-scale adjusts more rapidly for low frequencies and more slowly for high frequencies. The conversion between Mel(m) and Hertz(f) is defined as:

$$m = 2595 log_{10}(1 + f/700) \tag{3}$$

MFCC is another useful feature that reflects the static characteristics of sound signals. The extraction process gener-

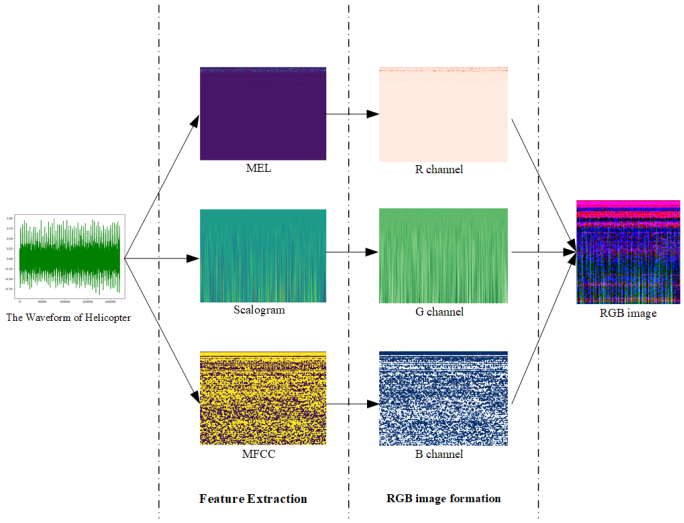

Fig. 1. Example: Picture on the far left is the waveform of helicopter sound. Pictures on the second col are MEL, scalogram and MFCC, respectively. The third col shows the relationship between extracted features and RGB channels(R channel corresponds to MEL, G channel corresponds to scalogram, B channel corresponds to MFCC). Picture on the far right is a RGB image fused by extracted features shown on the second col.

ally involves several steps: pre-emphasis, framing, windowing, Fourier Transform, passing through the mel filter bank, and Discrete Cosine Transform (DCT). DCT is often used in signal processing for data compression due to its strong energy concentration. This compression is lossy. MFCC can also be obtained by applying DCT to the mel spectrogram. In our experiment, both MEL and MFCC were extracted using the librosa package, a widely used Python library for music and audio signal analysis. The final feature is the scalogram, computed using continuous wavelet transform (CWT) [12]. CWT is a non-stationary spectral method, and the CWT of a signal $f(t)$ is defined as:

$$CWT(a,\tau) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} f(t)\Psi(\frac{t-\tau}{a})dt \tag{4}$$

where $CWT(a,\tau)$ is the wavelet transform coefficient for scale a at time τ. $f(t)$ is the origin signal and $\Psi(\tau)$ is the mother wavelet.In our experiment, scalograms were computed via a python package pywt and resized to 299 $\times$ 299.

As stated above, in our paper, sound signal is expressed in a format of RGB image. Hence, after the features are generated, these features will be converted into a RGB image, where R channel corresponds to MEL, G channel corresponds to MFCC, and B channel corresponds to scalogram. Fig. 1 shows the detailed process for three types of features(MEL, MFCC and scalogram) extraction and RGB image formation.

C. Model Training

1) Transfer Learning: In supervised learning, a large, well-annotated dataset is essential for achieving high model accuracy. However, collecting such datasets is challenging, particularly in environmental sound classification (ESC). The lack of suitable datasets is a major obstacle in ESC, which has led to growing interest in transfer learning. Transfer learning [13] enables a system to apply knowledge gained from previous tasks to new tasks.

There are two main approaches for transfer learning: layer freezing and model fine-tuning. One approach is to train a base network (pre-training model) and transfer its first k layers to the target network. This allows the remaining layers of the target network to be trained while leveraging the knowledge from the pre-training model, improving accuracy and reducing training time. Alternatively, fine-tuning can be used, where certain layers are frozen during training on the new task. However, if the target dataset is small and the model has many parameters, fine-tuning may lead to overfitting, negatively affecting performance. Therefore, fine-tuning is suitable only when the target dataset is large or the model has fewer parameters. Since ESC datasets are typically small, fine-tuning is not ideal in our case. To avoid overfitting and improve performance, our method adopts the first approach, using the pre-training model with frozen layers.

In general, pre-training model is first trained on the dataset of source domain and then its first k layers are copied to our network as the first k layers. We can denote the dataset of source domain as $D^{(S)} = \{d_m^{(S)} | m = 1, 2, ..., M^{(S)}\}$, where $d_m^{(S)}$ is the data instance and M is the size of the dataset. Besides, the corresponding class label of the source domain data can be represent as $Y^{(S)} = \{y_m^{(S)} | m = 1, 2, ..., M^{(S)}\}$. Similarly, the dataset of target domain and the corresponding class label can be denoted as $D^{(T)} = \{d_m^{(T)} | m = 1, 2, ..., M^{(T)}\}$ and $Y^{(T)} = \{y_m^{(T)} | m = 1, 2, ..., M^{(T)}\}$, respectively. The process of transferring knowledge from the source domain to the target domain can be expressed as follows:

$$O_i^P = \mathcal{N}_S^k(d_i^{(T)}),\ d_i^{(T)} \in D^{(T)}\ (i = 1, 2, ...M^{(S)}) \tag{5}$$

$$O_i^T = \mathcal{N}_T(O_i^P) \tag{6}$$

where $\mathcal{N}_S^k$ is the pre-training model chosen for transfer learning and its first k layers are frozen. O_i^P is the output of the pre-training model predicted by the ith instance of the target dataset. $\mathcal{N}_T$ is the CNN we designed and O_i^T is the output of our CNN. O_i^T denotes the predicted probabilities that i instance is of different class. The shape of O_i^T is $(C \times 1)$ where C denotes the number of categories. First of all, the first k layers of $\mathcal{N}_S^k$ are frozen and the remaining layers are trained on the source dataset. After training, we feed the target dataset into this pre-training model and get the output of $\mathcal{N}_S^k$ is O_i^P. In our experiment, the shape of O_i^P is (6×6×2048). Then, those outputs O_i^P are used to train our proposed convolutional neural network $\mathcal{N}_T$. During the training process, categorical cross entropy $\mathcal{L}$ is selected to evaluate our CNN performance and adjust the training direction of our model. $\mathcal{L}$ can be calculated as:

$$\mathcal{L} = -\sum_{i=1}^{M}\sum_{c=1}^{C} t_{i,c} ln[p(O_{i,c}^T)] \tag{7}$$

$p(O^T_{i,c})$ denotes the predicted probability that ith instance is of class c. For ith instance, only when class c is the correct classification, the $t_{i,c}$ is equal 1. In other case, $t_{i,c}$ is equal 0.

2) Xception: In our experiment, we use Xception as the pre-training model for CNN-based transfer learning. Xception, built entirely on depth-wise separable convolution layers, is an enhanced version of the Inception module [14]. Compared to Inception, Xception uses model parameters more efficiently, making it faster and more effective on ImageNet and JFT datasets. Xception applies depth-wise separable convolutions to reduce network complexity and decrease the number of training parameters. This improvement allows the network to be deeper and wider, achieving better performance.

In traditional convolution, a single step is used to extract features. For example, assuming M feature maps of size (R×C), N convolution kernels of size (3×3×M) are used to produce the final feature maps of size (R×C×N). Depth-wise separable convolution breaks this into two steps. First, M convolution kernels of size (3×3) are applied to the M input feature maps (R×C×M) individually. This produces M intermediate feature maps (R×C×1). Then, N convolution kernels of size (1×1) are used to combine the M intermediate results, yielding the final feature maps (R×C×N). This method uses fewer parameters while achieving the same result as traditional convolution. Therefore, depth-wise separable convolutions lead to a more efficient and flexible network, allowing for deeper and wider architectures with better performance. This is why we choose Xception as our pre-training model.

Additionally, Xception, pre-trained on the ImageNet dataset, is available through Keras, a popular Python package. This makes implementing Xception for transfer learning straightforward. Thus, we chose Xception as the pre-training model for our CNN-based transfer learning approach.

3) CNN: Our proposed CNN is inspired by VGG networks [15], which achieved the highest accuracy on the ILSVRC classification task and is widely used in image recognition systems. Unlike AlexNet, which uses large convolutional layers (11×11, 7×7, or 5×5), VGG employs smaller 3×3 convolutional layers stacked together. Using smaller filters allows the network to be deeper, enabling it to learn more complex patterns and improve classification performance.

In short, VGG replaces a 7×7 convolutional layer with three 3×3 layers and a 5×5 layer with two 3×3 layers, which enhances the network's depth and performance. The architecture is consistent across VGG-13, VGG-16, and VGG-19, with 3×3 convolutional layers and 2×2 max pooling layers.

Our CNN consists of five convolutional blocks, similar to VGG-16 [15]. Each block includes two 3×3 convolutional layers and one 1×1 convolutional layer. We apply padding of 1 pixel for each 3×3 convolution. Batch normalization (BN) is used after each 3×3 layer to speed up training, followed by a Rectified Linear Unit (ReLU) activation function. Between the two 3×3 convolutional layers, we include a 1×1 convolutional layer, inspired by the "Network in Network" (NiN) architecture [16]. The 1×1 layer in our CNN is not followed by a non-linearity, serving as a linear projection to increase network depth and enhance its ability to learn complex features. In addition, after all convolutional operations in each convolutional block, average-pooling is performed over a 2×2 window, with stride 2. With the help of average pooling layer, the advanced semantic information can be obtained, which is generally helpful for our classifier to classify. The structure of each convolutional block is shown in Fig. 2.

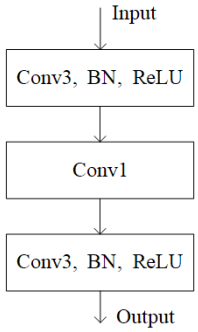

Fig. 2. The structure of convolutional block: "Conv3" represents 3×3 convolutional layer with the filter size of 3×3. "BN" represents the Batch Normalization layer. "ReLU" is the activation function used for our CNN. "Conv1" represents 1×1 convolutional layer with the filter size of 1×1.

TABLE I
THE CNN ARCHITECTURE

Layer	Data Shape
Input	(36, 2048, 3)
Conv3-32, BN, ReLU Conv1-32 Conv3-32, BN, ReLU	(36, 2048, 32)
Average Pooling	(18, 1024, 32)
Conv3-64, BN, ReLU Conv1-64 Conv3-64, BN, ReLU	(18, 1024, 64)
Average Pooling	(9, 512, 64)
Conv3-128, BN, ReLU Conv1-128 Conv3-128, BN, ReLU	(9, 512, 128)
Average Pooling	(4, 256, 128)
Conv3-256, BN, ReLU Conv1-256 Conv3-256, BN, ReLU	(4, 256, 256)
Average Pooling	(2, 128, 256)
Conv3-256, BN, ReLU Conv1-256 Conv3-256, BN, ReLU	(2, 128, 256)
Average Pooling	(1, 64, 256)
Global Max Poolinf	(1, 256)
Dropout	(1, 256)
FC	(1, # Number of classes)

[1] "Conv3-32" represents using 32 channels with filter size of (3×3) for convolution.
[2] "Conv1-32" represents using 32 channels with filter size of (1×1) for convolution.
[3] The Pooling filter size of average pooling layer is (2×2).

At the end of each convolutional block, we apply a global max-pooling layer and a dropout layer. The global max-pooling layer focuses on capturing relationships between

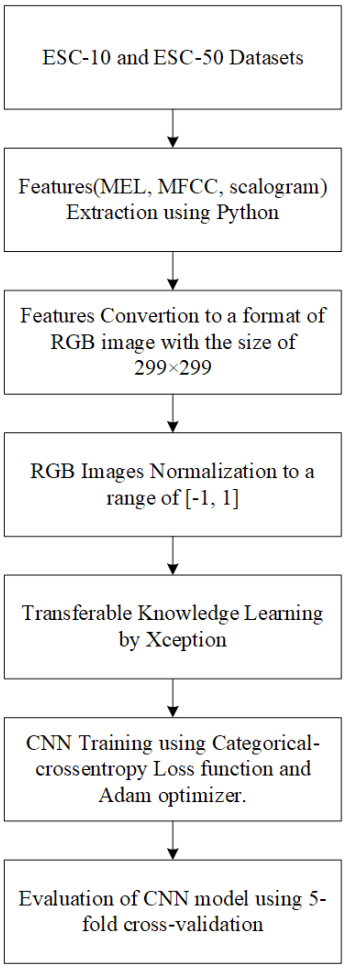

Fig. 3. The workflow procedure

different channels, producing vectors from high-level feature maps. To prevent overfitting, we use a dropout layer [17] with a probability of 0.5. Finally, a fully connected layer followed by a softmax function computes the probability for each sound class and performs classification. The complete architecture of our proposed CNN is summarized in Table I.

III. Experiment and Results

A. Experiment settings

The workflow of our classification system is shown in Fig. 3. We use the ESC-50 (2000 recordings, 50 classes) and ESC-10 (400 recordings, 10 classes) datasets for evaluating our proposed CNN. Both datasets are pre-sorted into 5 folds for cross-validation, allowing for precise evaluation.

The first step in our experiment is feature extraction using Python packages such as librosa, OpenCV, and pywt. MEL features are extracted with parameters set as Hanning window, 2048 size, 22050 Hz sample rate, and 512 hop length. MFCC is calculated using librosa functions directly from the audio signal. Scalogram extraction involves two steps: first, we apply continuous wavelet transform (CWT) from the pywt package, and then resize the output to 299×299 using OpenCV.

TABLE II
THE OVERALL COMPARE OF PERFORMANCE ON ESC-10 AND ESC-50

Model	ESC-10	ESC-50
SVM [8]	67.5%	39.6%
KNN [8]	66.7%	32.3%
Random Forest [8]	72.7%	44.3%
PiczakCNN [19]	73%	64.5%
SoundNet [6]	92.2%	74.2%
Our Proposed CNN	**93.3%**	**84.5%**
Human Performance [19]	96%	81%

Next, we convert MEL, MFCC, and scalogram features into RGB images (299×299), normalizing all images to the range of [-1,1]. This results in 2000 normalized RGB images for ESC-50 and 400 for ESC-10. These images are used to train our CNN based on transfer learning, with Xception as the pre-trained model. The training process uses a batch size of 32, an initial learning rate of 0.0001, and a decay of 0.000001. We employ the Adam optimizer [18] and train the model for multiple epochs.

B. Results

Table 2 compares our proposed CNN with previous methods for ESC on the ESC-10 and ESC-50 datasets. Our CNN achieves 93.3% accuracy on the ESC-10 dataset and 84.5% on the ESC-50 dataset, outperforming other systems. On ESC-50, our approach improves accuracy by 44.9%, 52.2%, 40.2%, 20%, and 10.3% compared to other systems. On ESC-10, our method also surpasses others, with improvements of 20.3% over PiczakCNN and 1.1% over SoundNet. Human performance is 96% on ESC-10 and 81% on ESC-50. While our CNN performs better than human classification on ESC-50, it is slightly inferior to human performance on ESC-10. This is because the ESC-10 dataset has only 10 categories, which makes it easier for humans to classify sounds [8]. As the number of categories increases, automatic systems outperform humans due to the added complexity.

In our experiment, CNN training is done with different epochs. The accuracy curve of different epochs is shown in Fig. 4(ESC-50) and Fig.5(ESC-10). Fig. 4 shows that as the number of epochs increases, the system's accuracy also increases, peaking at 35 epochs. After 35 epochs, the accuracy starts to decline. A similar trend is observed in Fig. 5, where the accuracy increases until 230 epochs and decreases thereafter. The highest accuracy for the ESC-10 system is achieved with 230 epochs, reaching 93.3

CNNs typically outperform traditional classification methods like SVM, KNN, and HMM in learning and classification tasks. However, overfitting is a common issue when training CNNs on limited datasets or with too many epochs. To address this, we apply three strategies: transfer learning, a dropout layer with 50% probability, and choosing an optimal number of training epochs. Transfer learning and the dropout layer are detailed earlier, while the optimal epochs are shown in Fig. 4 and Fig. 5. For ESC-50, training with 35 epochs yields the highest accuracy (84.5%), and for ESC-10, 230 epochs give

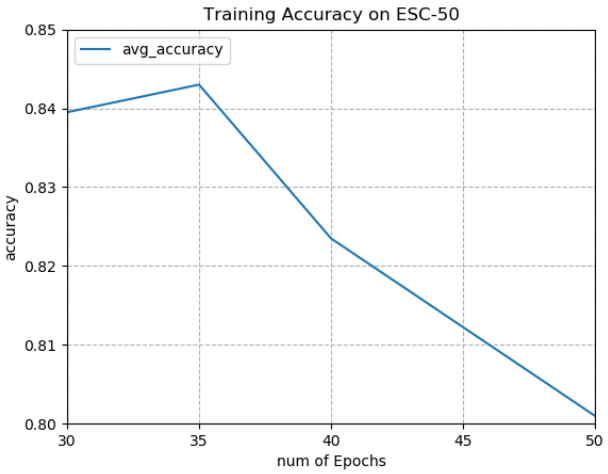

Fig. 4. The accuracy curve of proposed CNN training on ESC-50 with different epochs

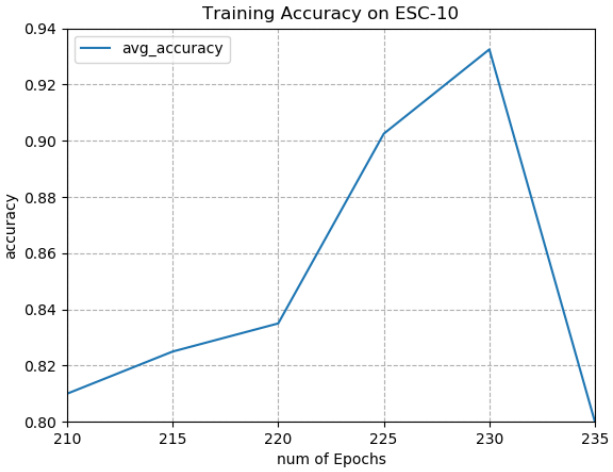

Fig. 5. The accuracy curve of proposed CNN training on ESC-10 with different epochs

the best result (93.3%). Beyond these points, the model suffers from overfitting, causing accuracy to drop.

Overall, our experimental results demonstrate that the proposed approach is effective, achieving better performance than Piczak's method (64.5% and 73% on ESC-50 and ESC-10, respectively). By selecting the appropriate number of training epochs, our CNN achieves 84.5% accuracy on ESC-50 and 93.3% on ESC-10.

IV. Conclusion

In this paper, we proposed a convolutional neural network (CNN) using transfer learning (with Xception as the pre-trained model) for environmental sound classification (ESC). We selected three widely used features—MEL, MFCC, and scalogram—to represent the environmental sounds. After extracting these features, we converted them into RGB images of size 299 × 299 (required input size for Xception) to train our CNN. After training, our model achieved high accuracy, outperforming other state-of-the-art models on the ESC-10 and ESC-50 datasets. The proposed CNN can also be applied to other sound classification tasks that utilize transfer learning and deep learning techniques.

References

[1] S. D. Mainkar and S. P. Mahajan, "EMD based efficient discrimination of real-world environmental sounds using SVM classifier," *2015 International Conference on Information Processing (ICIP)* , Pune, 2015, pp. 272-277.

[2] Jia-Ching Wang, Jhing-Fa Wang, Kuok Wai He and Cheng-Shu Hsu, "Environmental Sound Classification using Hybrid SVM/KNN Classifier and MPEG-7 Audio Low-Level Descriptor," *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, Vancouver, BC, 2006, pp. 1731-1735.

[3] O. K. Toffa and M. Mignotte, "Environmental Sound Classification Using Local Binary Pattern and Audio Features Collaboration," *IEEE Trans. Multimedia* , vol. 23, pp. 3978-3985, 2021.

[4] P. Mayorga, C. Druzgalski, J. Miranda, V. Zeljkovic and O. H. González, "The HMM diagnostic models of respiratory sounds," *2014 Pan American Health Care Exchanges (PAHCE)*, Brasilia, 2014, pp. 1-6.

[5] Jwu-Sheng Hu, Chieh-Cheng Cheng and Wei-Han Liu, "Robust speaker's location detection in a vehicle environment using GMM models," *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 36, no. 2, pp. 403-412, April 2006.

[6] Y. Aytar, C. Vondrick, and A. Torralba, "SoundNet: Learning sound representations from unlabeled video," *in Proc. Conf. Neural Inf. Process. Syst.*, 2016, pp. 892–900.

[7] L. Nanni, G. Maguolo, S. Brahnam, and M. Paci, "An Ensemble of Convolutional Neural Networks for Audio Classification," *arXiv e-prints*, p. arXiv:2007.07966, July 2020.

[8] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," *The 23rd ACM international conference on Multimedia - MM'15*, pp. 1015-1018, 2015.

[9] T. Tran and J. Lundgren, "Drill Fault Diagnosis Based on the Scalogram and Mel Spectrogram of Sound Signals Using Artificial Intelligence," *IEEE Access*, vol. 8, pp. 203655-203666, 2020.

[10] B. Milner and X. Shao, "Prediction of Fundamental Frequency and Voicing From Mel-Frequency Cepstral Coefficients for Unconstrained Speech Reconstruction," *IEEE Trans. Audio Speech Lang. Process.*, vol. 15, no. 1, pp. 24-33, 2007.

[11] T. Virtanen, M. D. Plumbley and D. Ellis, "Computational analysis of sound scenes and events," Heidelberg: Springer, pp. 72-73, 2018.

[12] L. Cnockaert, P. -F. Migeotte, L. Daubigny, G. K. Prisk, F. Grenez and R. C. Sa, "A Method for the Analysis of Respiratory Sinus Arrhythmia Using Continuous Wavelet Transforms," *IEEE Trans. Biomed. Eng.*, vol. 55, no. 5, pp. 1640-1642, May 2008.

[13] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *arXiv e-prints*, p. arXiv:1411.1792, Nov. 2014.

[14] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, pp. 1800-1807, 2017.

[15] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv e-prints*, p. arXiv:1409.1556, Sept. 2014.

[16] M. Lin, Q. Chen, and S. Yan, "Network in network," CoRR, vol. abs/1312.4400, 2013. [Online]. Available: http://arxiv.org/abs/1312.4400

[17] X. Bouthillier, K. Konda, P. Vincent, and R. Memisevic, "Dropout as data augmentation," *arXiv e-prints*, p. arXiv:1506.08700, June 2015.

[18] D. P. Kingma and L. J. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations(ICLR)*, 2014

[19] " K. J. Piczak, "Environmental sound classification with convolutional neural networks," 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP), Boston, MA, pp. 1-6, 2015.