GENERAL-PURPOSE AUDIO TAGGING FROM NOISY LABELS USING CONVOLUTIONAL NEURAL NETWORKS

Turab Iqbal, Qiuqiang Kong, Mark D. Plumbley, Wenwu Wang

Centre for Vision, Speech and Signal Processing, University of Surrey
{t.iqbal, q.kong, m.plumbley, w.wang}@surrey.ac.uk

ABSTRACT

General-purpose audio tagging refers to classifying sounds that are of a diverse nature, and is relevant in many applications where domain-specific information cannot be exploited. The DCASE 2018 challenge introduces Task 2 for this very problem. In this task, there are a large number of classes and the audio clips vary in duration. Moreover, a subset of the labels are noisy. In this paper, we propose a system to address these challenges. The basis of our system is an ensemble of convolutional neural networks trained on log-scaled mel spectrograms. We use preprocessing and data augmentation methods to improve the performance further. To reduce the effects of label noise, two techniques are proposed: loss function weighting and pseudo-labeling. Experiments on the private test set of this task show that our system achieves state-of-the-art performance with a mean average precision score of 0.951.

Index Terms— Audio classification, convolutional network, recurrent network, deep learning, data augmentation, label noise

1. INTRODUCTION

Audio tagging is a classification problem that is concerned with categorizing audio clips based on the presence of sound events. These events could be domestic sounds such as a telephone ringing, outdoor sounds such as a car passing by, and anything else that may be relevant to an application. Historically, classifiers have relied on domain-specific techniques to achieve good performance, such as in speech recognition [1] and music information retrieval [2]. However, with new applications such as smart homes [3] and smart cities [4], there is growing interest in general-purpose audio classifiers.

The Detection and Classification of Acoustic Scenes and Events (DCASE) [5] is a recurring challenge with several tasks pertaining to audio classification. DCASE 2018 introduces Task 2 [6], which poses the problem of general-purpose audio tagging. This task uses a subset of the FSD dataset [7], and is comprised of 41 audio classes with labels from Google's AudioSet ontology [8]. As this is a high number of classes, it demands good discriminative abilities from the classifier. A training set of 9473 labeled examples is provided in order to use supervised learning methods. However, approximately $60\,\%$ of the labels are unverified. Of these unverified labels, at least $65\,\%$ to $70\,\%$ of the labels are correct. The presence of incorrectly-labeled examples can negatively affect training, so it is important that the learning algorithm is robust with respect to such examples. Another property of this dataset is that the duration of the audio clips varies from $0.3\,\mathrm{s}$ to $30\,\mathrm{s}$. This is a problem because many training models expect a fixed-length input.

In this paper, we propose a system to address these challenges. The system we develop is based on a number of convolutional neural networks trained on log-mel spectrograms. We investigate different architectures of convolutional neural networks and show that they complement each other by ensembling predictions using a technique called stacking. We also use preprocessing and data augmentation methods to improve the performance further. To reduce the effects of incorrect labels, two ideas are proposed: loss function weighting and pseudo-labeling.

This paper is organized as follows. Section 2 introduces related work. Section 3 details the preprocessing and feature extraction methods. Section 4 proposes the convolutional neural networks and training methodology. Section 5 presents the experiments and results. Lastly, Section 6 concludes with a summary.

2. RELATED WORKS

Deep neural networks have recently become a popular choice for audio classification due to their leading performance in many tasks, including general-purpose audio tagging [9]. Most, if not all, of the neural network architectures that achieve state-of-the-art results are based on convolutional neural networks (CNN) [10, 11, 12]. Hybrid architectures such as convolutional recurrent neural networks (CRNN) have also been adopted with great success [13, 14, 15], but more for tasks that also require localization. Given a training model, an effective way to improve the performance further is to use data augmentation, as observed in [16] and [17]. We use both CNNs and CRNNs with data augmentation, but also consider label noise.

Learning from noisy labels is a problem that has several decades of research behind it [18]. It was shown in [19] that neural networks can converge to near-zero loss on training sets even when the labels are completely random. This is problematic because it suggests that incorrect labels can have a significant impact on the generalization performance of a neural network. One way to mitigate the effects of label noise is to apply a correction to the loss function as in [20], which requires estimating a noise transition matrix. In [21], this is done by incorporating an additional layer in the neural network. We also modify the loss function, but make simpler assumptions and hence propose a simpler method. Furthermore, we combine this with a technique called pseudo-labeling.

3. PREPROCESSING AND FEATURE EXTRACTION

3.1. Silence Removal

In the dataset of Task 2, we observed that there is little background noise present in the audio clips. However, some of the clips contain segments of silence. Long sequences of silence are not characteristic of the sounds themselves, and only indicate the start or end of sounds. For this reason, we include in our pipeline an algorithm to extract the non-silent segments of the audio signal. This discards the silence and means that separate segments can be considered as separate inputs.

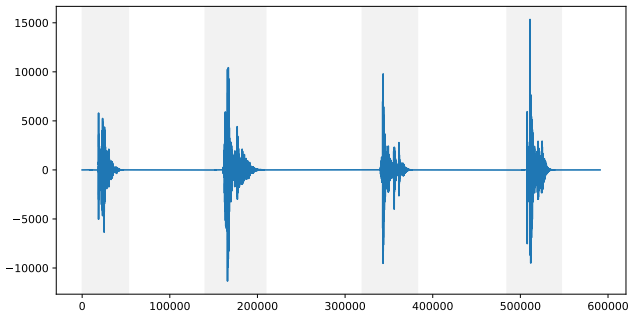

Figure 1: Illustration of the silence removal process on the file "071e836c.wav", which is 13 s in length. The non-silent segments that are extracted are highlighted in gray. These segments would be considered as separate inputs.

Table 1: Log-mel spectrogram parameters

Parameter	Configuration A	Configuration B
Sample rate	32 000 Hz	32 000 Hz
Window size	1024	512
Hop size	512	256
Mel bands	64	64

To detect silence, we segment the audio signal into frames and threshold each frame's root mean square (RMS) energy. We define "silence" to include very quiet background noise, so it does not have to match the threshold of hearing. In our algorithm, a non-silent segment is the span of non-silent frames that are within some proximity (they do not have to be contiguous), and also includes some excess silence to prevent over-cropping. This is exemplified in Figure 1, where four segments are highlighted.

3.2. Feature Extraction

Following silence removal, the extracted segments are considered as separate inputs. These are downsampled to 32 kHz and transformed into log-scaled mel-frequency (log-mel) spectrograms. Log-mel features have been shown to outperform traditional representations such as mel-frequency cepstrum coefficients (MFCCs) in modern neural network architectures [22], which benefit from the additional information and do not need de-correlated inputs [23].

We use two sets of parameters to extract the log-mel features – a standard configuration and a narrow-band configuration – the values of which are given in Table 1. We selected these values based on experiments on a validation set. In our experiments, these configurations produced complementing results. Indeed, the different resolutions appear to capture different characteristics.

After feature extraction, each feature vector is partitioned into chunks of a fixed size, which resolves the problem of varying clip durations. When the length of the feature vector is less than the chunk length, it is padded. When it is greater, but not evenly divisible by the chunk length, an additional chunk is added to align with the end of the feature vector so that it includes the remainder.

The chunk size is an important parameter, as a chunk that is too short may not encompass a sound in its entirety. On the other hand, a chunk that is too long will mostly contain padding data for short audio segments. We choose a chunk size of 128×64, where the first axis is the temporal dimension. This corresponds to 2 s chunks and 1 s chunks for configurations A and B, respectively (cf. Table 1). Approximately 80 % of the segments in the training set are more than 1 s in duration, and 60 % are more than 2 s.

4. TRAINING AND INFERENCE

In order to utilize the training set that is provided for this task, we use two types of neural networks: CNNs and CRNNs. For each type, we also use two variants: one using standard convolutions and another using gated convolutions. Since there are two log-mel configurations, this gives eight training models in total. In the subsections to follow, we describe the architectures, learning with label noise, data augmentation, and ensembling.

4.1. Neural Network Architectures

The neural network architectures are outlined in Table 2. Beginning with the standard CNN, it is essentially equivalent to the "VGG13" network proposed in [24], hence the name. Each convolutional block consists of two convolutional layers followed by a max pooling layer that halves each spatial dimension. The convolutional layers use a ReLU activation function [25] as well as batch normalization [26] as a form of regularization. After the convolutional blocks, global average pooling is applied, i.e. each feature map is averaged across both dimensions. Finally, a densely-connected softmax layer is used to generate the predictions.

The CRNN architecture is an extension of VGG13. Instead of applying global average pooling after the convolutions, only the frequency dimension is averaged so that temporal information is preserved. A bidirectional recurrent layer [27] is then applied to output a vector, $\mathbf{s}_t$, for each time step $t \in [1, T]$. Averaging these vectors gives $\mathbf{s} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{s}_t$, which is the output prior to the softmax layer. The motivation for using a recurrent layer is to learn the temporal dynamics of the input [13]. In our experiments, this improved the overall performance by up to 0.5 %.

The other two architectures are GCNN and GCRNN, which are variants of VGG13 and CRNN, respectively. The difference is that each convolutional layer is replaced with a gated convolutional layer [28]. The idea of a gated layer is inspired by the gating mechanisms found in recurrent neural networks [29, 30], and is used to control the information that is propagated to deeper layers. This mechanism has been shown to produce good results for similar tasks [15].

4.2. Learning from Noisy Labels

The presence of unverified labels poses a problem for learning, as neural networks are susceptible to overfitting on incorrectly-labeled examples [19]. On the other hand, training with verified examples *only* means that a large number of unverified labels that are otherwise correct are discarded. Our tests showed that performance dropped by up to 5 % when a model was trained on verified examples only. Therefore, we propose two techniques.

The first technique is to weight the training loss function such that its magnitude is lowered for unverified examples. The rationale is that if an example is incorrectly labeled, the computed loss will be incorrect and should be disregarded. Of course, we do not know whether it is correct or not if the label is unverified. Let $\eta \in (0, 1)$ be the weight applied to unverified examples. Given a loss function, $L(\mathbf{y}, \hat{\mathbf{y}})$, the weighted loss function is then given by

$$\tilde{L}(\mathbf{y}, \hat{\mathbf{y}}) \coloneqq (\eta \cdot \mathbb{1}_{\mathcal{X}_N}(\mathbf{x}) + \mathbb{1}_{\mathcal{X}_N^c}(\mathbf{x}))L(\mathbf{y}, \hat{\mathbf{y}}), \qquad (1)$$

Table 2: Description of the neural networks. Each convolutional layer uses a receptive field size of 3, as in [24]. "GLU" refers to the use of gated linear units, as described in [28]. "Bi-GRU" refers to using bidirectional gated recurrent units [30].

Feature Size	CRNN	GCRNN
128×64	Log-mel spectrogram	
	$2 \times \{\text{conv } 64, \text{ReLU}\}$	$2 \times \{\text{conv } 64, \text{GLU}\}$
64×32	2×2 Max Pooling	
	$2 \times \{\text{conv } 128, \text{ReLU}\}$	$2 \times \{\text{conv } 128, \text{GLU}\}$
32×16	2×2 Max Pooling	
	$2 \times \{\text{conv } 256, \text{ReLU}\}$	$2 \times \{\text{conv } 256, \text{GLU}\}$
16×8	2×2 Max Pooling	
	$2 \times \{\text{conv } 512, \text{ReLU}\}$	$2 \times \{\text{conv } 512, \text{GLU}\}$
8×4	2×2 Max Pooling	
	$2 \times \{\text{conv } 512, \text{ReLU}\}$	$2 \times \{\text{conv } 512, \text{GLU}\}$
4×2	2×2 Max Pooling	
	Bi-GRU, 512, ReLU (optional)	
	Global Average Pooling	
	Softmax (41 Classes)	

where $\mathcal{X}_N$ is the subset of the training set that is unverified and $\mathcal{X}_N^c$ is the complement. The parameter η can be considered as the confidence that the unverified labels are correct. It can be be determined using a validation set or set to 1 minus the noise rate.

The second technique, known as pseudo-labeling, is to relabel the unverified examples prior to training using a previously-trained classifier. For pseudo-labeling to be effective, the error rate of the classifier should be lower than the noise rate of the original labels, because the error rate can be considered as the new noise rate. The previously-trained classifier in this case is just the same system described in this paper but without pseudo-labeling.

Another form of pseudo-labeling that we propose is to "promote" examples from unverified to verified by corroborating the label with the predictions of the previously-trained classifier. The examples if promoted if the classifier prediction agrees with confidence greater than τ. The confidence threshold, τ, should be high enough that the false positive rate of the classifier is low.

4.3. Data Augmentation

When training a neural network, overfitting is a common problem in which the network learns to predict the training examples with very high accuracy but cannot generalize to new data. This is likely to occur with smaller datasets, and was found to be the case with the dataset of Task 2 by verifying on a validation set. To prevent this, data augmentation is a popular approach [31] that increases the size of the training set without manual intervention. In our work, we use a method called *mixup* [32] to create new examples.

Mixup is a method that generates new data during training by randomly mixing pairs of inputs and their associated target values. Consider a pair of inputs, $\mathbf{x_1}$ and $\mathbf{x_2}$, and their one-hot-encoded target values, $\mathbf{y_1}$ and $\mathbf{y_2}$. To mix these, a parameter, $\lambda \in (0, 1)$, is used to create convex combinations.

$$\mathbf{x} = \lambda \mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2. \tag{2}$$

$$\mathbf{y} = \lambda \mathbf{y}_1 + (1 - \lambda)\mathbf{y}_2. \tag{3}$$

The output, $(\mathbf{x}, \mathbf{y})$, is then used as the training example rather than the original examples. If loss function weighting is used, $w(\mathbf{x}) \coloneqq \eta \cdot \mathbb{1}_{\mathcal{X}_N}(\mathbf{x}) + \mathbb{1}_{i\mathcal{X}_N^c}(\mathbf{x})$ should also be mixed.

Table 3: Training parameters

Parameter	Value
Batch size	128
Learning rate (LR)	0.0005
LR decay factor	0.9
LR decay rate	2

In the original paper [32], a different value of λ is used for each mini-batch by sampling from a beta distribution, $\mathrm{B}(\alpha, \alpha)$. The hyperparameter α controls the distribution's shape, where a value of 1.0 reduces it to a uniform distribution. A lower value of α will be more likely to produce values of λ that are closer to 0 and 1, which weakens the effect of mixup.

Mixup has been used before for audio classification and has been shown to be beneficial [17]. We ultimately followed the method used in [32], but also experimented with a variation. This involves constraining some or all of the example pairs to belong to the same class. Unfortunately, this did not improve the performance of our system and actually worsened it in the extreme case of constraining all the pairs. This suggests that mixing inter-class examples is an important aspect of mixup's success.

5. EXPERIMENTS

5.1. System Setup

To train the various models, the training set was split into five cross-validation folds, ensuring that there was a similar number of verified examples in each fold. The cross-entropy function was used as the training loss and Adam [33] was used as the gradient descent algorithm. Refer to Table 3 for the values of the hyperparameters. Decay rate is the number of epochs until the learning rate is decayed.

For loss function weighting, we used a weight of $\eta = 0.7$ when pseudo-labeling was not used (to determine the pseudo-labels in the first place) and $\eta = 0.9$ when it was. When promoting labels, we used a confidence threshold of $\tau = 0.7$. For mixup, the parameter α was set to 1.0. All of the hyperparameters were selected based on evaluations on a validation set containing verified labels only. This includes the parameters in Table 3 too.

In terms of generating the predictions, the top four epochs were selected based on performance on the validation set. The metric used was the mean average precision (MAP) score. Recalling that the inputs to the neural networks are chunks, and that the chunks are from sections of the original audio clip (cf. Section 3), the chunk predictions need to be merged to produce clip-level predictions. This was achieved using the geometric mean, as this is less sensitive to outliers than the arithmetic mean. With the clip-level predictions, the top four epochs were merged using the arithmetic mean.

5.2. Ensembling

To combine the predictions of the different models, we used an ensembling method known as stacking [34, 35, 36]. In this method, the base model predictions are used as features to train a second-level classifier. The output of a base model is an $N \times K$ vector of probabilities, where N is the number of data samples and $K = 41$ is the number of classes. By concatenating the outputs of the models, the result is an $N \times 8K$ vector; this is the input of the new classifier.

Table 4: Training set results. Only the results of configuration A models are given to highlight the difference in architectures.

Model	MAP@3
VGG13	0.950
GCNN	0.951
CRNN	0.952
GCRNN	0.958
Arithmetic Mean	0.968
Stacking	0.972

Table 5: Test set results comparing the 8-model stacked ensemble with a 4-model version that excludes VGG13 and CRNN models. The results of the competition's baseline system is also included.

Model	Private	Public	All
Baseline	0.694	0.704	–
4-Model Stacking	0.948	0.956	0.950
8-Model Stacking	0.951	0.961	0.953

As the validation sets constitute the training set, the validation set predictions were used to generate the features for the training set. Similarly, the test set predictions were used as the test set features. We used logistic regression with an L_2 penalty as the second-level classifier. It was configured to use class weights to compensate for class imbalance and sample weights as described in Section 4.2.

5.3. Results

To assess the performance of our system, we evaluated the training set and test set predictions. It was possible to evaluate the training set predictions because we used cross-validation folds. We only evaluated the manually-verified training examples. The test set was split into a public set and a private set by the challenge organizers. Therefore, we report results for both sets individually and also when the two are combined. The metric used to assess the performance is the mean average precision (MAP@3) score, which is defined as

$$\mathrm{MAP@3} = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{\min\{K,3\}} P(i), \tag{4}$$

where N is the number of data samples, $K = 41$ is the number of classes, and $P(i)$ is the precision at cutoff i.

The results for the training set are shown in Table 4. The systems that are compared are the single models (log-mel configuration A only), an arithmetic-mean ensemble of the models, and the stacked ensemble described in the previous section. It can be seen that the mean ensemble performs much better than all of the single models – by almost 2 %. However, the stacked ensemble performs the best, with a MAP@3 score of 0.972. The weight-learning capability of stacking, with respect to model and class, appears to help.

In Table 5, the results for the test set are presented. We look at the 8-model stacked ensemble compared to a smaller 4-model version. In the latter, the VGG13 and CRNN architectures are omitted. The results of both systems are far superior to the competition's baseline system. Although the additional models in the 8-model version help, the difference is minor. This can be explained by the lack of diversity that the omitted models have to offer.

6. CONCLUSION

This paper described an approach to audio tagging in which the audio signals to be classified were of a diverse nature. The approach was based on our efforts in Task 2 of the DCASE 2018 challenge. The challenges of this task include audio clips of varying duration and incorrectly-labeled training examples. Our method involved preprocessing the audio, extracting log-mel feature vectors, and partitioning the feature vectors into fixed chunks to be considered as separate inputs. To train on these inputs, a number of convolutional and convolutional-recurrent neural networks were introduced. We used mixup for data augmentation. Several techniques were also used to resolve the problem of incorrect labels, including pseudo-labeling and loss function weighting. In evaluating our system on the DCASE 2018 Task 2 Kaggle private test set, we achieved a mean average precision score of 0.951, placing us in 3rd place out of 558 in the Kaggle private leaderboard.

7. ACKNOWLEDGMENT

This research was supported by the EPSRC grant EP/N014111/1, "Making Sense of Sounds", and a Research Scholarship from the China Scholarship Council (CSC), No. 201406150082. We would also like to thank Yong Xu for his contributions in the early stages of the competition and the relevant work he did in previous challenges.

8. REFERENCES

[1] J. H. L. Hansen and T. Hasan, "Speaker recognition by machines and humans: A tutorial review," *IEEE Signal Process. Mag.*, vol. 32, no. 6, pp. 74–99, 2015.

[2] M. Schedl, E. Gómez, and J. Urbano, "Music information retrieval: Recent developments and applications," *Found. Trends Inf. Retr.*, vol. 8, no. 2-3, pp. 127–261, 2014.

[3] S. Krstulović, "Audio event recognition in the smart home," in *Computational Analysis of Sound Scenes and Events*, 1st ed., T. Virtanen, M. D. Plumbley, and D. Ellis, Eds. Cham: Springer, 2018, pp. 335–371.

[4] J. P. Bello, C. Mydlarz, and J. Salamon, "Sound analysis in smart cities," in *Computational Analysis of Sound Scenes and Events*, 1st ed., T. Virtanen, M. D. Plumbley, and D. Ellis, Eds. Cham: Springer, 2018, pp. 373–397.

[5] D. Giannoulis, E. Benetos, D. Stowell, M. Rossignol, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: An IEEE AASP challenge," in *IEEE Workshop Applied Signal Processing Audio and Acoustics (WASPAA)*, New Paltz, NY, 2013, pp. 1–4.

[6] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of Freesound audio with AudioSet labels: Task description, dataset, and baseline," *arXiv preprint arXiv:1807.09901*, 2018.

[7] E. Fonseca, J. Pons, X. Favory, F. Font, D. Bogdanov, A. Ferraro, S. Oramas, A. Porter, and X. Serra, "Freesound datasets: A platform for the creation of open audio datasets," in *Proc. 18th Int. Society Music Information Retrieval Conf. (ISMIR)*, Suzhou, China, 2017, pp. 486–493.

[8] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio Set: An ontology and human-labeled dataset for audio

events," in *IEEE Int. Conf. Acoustic Speech and Signal Processing (ICASSP)*, New Orleans, LA, 2017, pp. 776–780.

[9] A. Mesaros, T. Heittola, E. Benetos, P. Foster, M. Lagrange, T. Virtanen, and M. D. Plumbley, "Detection and classification of acoustic scenes and events: Outcome of the DCASE 2016 challenge," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 26, no. 2, pp. 379–393, 2018.

[10] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *IEEE 25th Int. Workshop Machine Learning Signal Processing (MLSP)*, Boston, MA, 2015, pp. 1–6.

[11] H. Eghbal-zadeh, B. Lehner, M. Dorfer, and G. Widmer, "A hybrid approach with multi-channel I-vectors and convolutional neural networks for acoustic scene classification," *arXiv preprint arXiv:1706.06525*, 2017.

[12] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *IEEE Int. Conf. Acoustic Speech and Signal Processing (ICASSP)*, New Orleans, LA, 2017, pp. 131–135.

[13] E. Çakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 25, no. 6, pp. 1291–1303, 2017.

[14] E. Çakır, S. Adavanne, G. Parascandolo, K. Drossos, and T. Virtanen, "Convolutional recurrent neural networks for bird audio detection," in *Proc. European Signal Processing Conf. (EUSIPCO)*, Kos, Greece, 2017, pp. 1744–1748.

[15] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, "Large-scale weakly supervised audio classification using gated convolutional neural network," in *IEEE Int. Conf. Acoustic Speech and Signal Processing (ICASSP)*, Calgary, Canada, 2018, pp. 121–125.

[16] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Process. Lett.*, vol. 24, no. 3, pp. 279–283, 2017.

[17] K. Xu, D. Feng, H. Mi, B. Zhu, D. Wang, L. Zhang, H. Cai, and S. Liu, "Mixup-based acoustic scene classification using multi-channel convolutional neural network," in *Advances Multimedia Information Processing (PCM)*, R. Hong, W.-H. Cheng, T. Yamasaki, M. Wang, and C.-W. Ngo, Eds., 2018, pp. 14–23.

[18] B. Frenay and M. Verleysen, "Classification in the presence of label noise: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 5, pp. 845–869, 2014.

[19] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *Proc. 5th Int. Conf. Learning Representations (ICLR)*, New Orleans, LA, 2017.

[20] G. Patrini, A. Rozza, A. Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, 2017, pp. 2233–2241.

[21] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, and R. Fergus, "Training convolutional networks with noisy labels," in *Proc. 3rd Int. Conf. Learning Representations (ICLR)*, San Diego, CA, 2015.

[22] M. Huzaifah, "Comparison of time-frequency representations for environmental sound classification using convolutional neural networks," *arXiv preprint arXiv:1706.07156*, 2017.

[23] L. Deng, J. Li, J. Huang, K. Yao, D. Yu, F. Seide, M. L. Seltzer, G. Zweig, X. He, J. D. Williams, Y. Gong, and A. Acero, "Recent advances in deep learning for speech research at microsoft," in *IEEE Int. Conf. Acoustic Speech and Signal Processing (ICASSP)*, Vancouver, Canada, 2013, pp. 8604–8608.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learning Representations (ICLR)*, San Diego, CA, 2015.

[25] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Machine Learning (ICML)*, Haifa, Israel, 2010, pp. 807–814.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Machine Learning (ICML)*, ser. Proc. Mach. Learn. Res., vol. 37, Lille, France, 2015, pp. 448–456.

[27] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, 1997.

[28] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. 34th Int. Conf. Machine Learning (ICML)*, ser. Proc. Mach. Learn. Res., vol. 70, Sydney, Australia, 2017, pp. 933–941.

[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[30] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances Neural Information Processing Systems (NIPS)*, Lake Tahoe, NV, 2012, pp. 1097–1105.

[32] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Proc. 6th Int. Conf. Learning Representations (ICLR)*, Vancouver, Canada, 2018.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learning Representations (ICLR)*, San Diego, CA, 2015.

[34] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, 1992.

[35] K. M. Ting and I. H. Witten, "Issues in stacked generalization," *J. Artif. Int. Res.*, vol. 10, no. 1, pp. 271–289, 1999.

[36] M. J. van der Laan, E. C. Polley, and A. E. Hubbard, "Super learner," *Stat. Appl. Genet. Mol. Biol.*, vol. 6, no. 1, 2007.