TRAINING GENERAL-PURPOSE AUDIO TAGGING NETWORKS WITH NOISY LABELS AND ITERATIVE SELF-VERIFICATION

Matthias Dorfer

Johannes Kepler University
Institute of Computational Perception
Linz, 4040, Austria
matthias.dorfer@jku.at

Gerhard Widmer

Johannes Kepler University
Institute of Computational Perception
Linz, 4040, Austria
gerhard.widmer@jku.at

ABSTRACT

This paper describes our submission to the first Freesound general-purpose audio tagging challenge carried out within the DCASE 2018 challenge. Our proposal is based on a fully convolutional neural network that predicts one out of 41 possible audio class labels when given an audio spectrogram excerpt as an input. What makes this classification dataset and the task in general special, is the fact that only 3,700 of the 9,500 provided training examples are delivered with manually verified ground truth labels. The remaining non-verified observations are expected to contain a substantial amount of label noise (up to 30-35% in the "worst" categories). We propose to address this issue by a simple, iterative self-verification process, which gradually shifts unverified labels into the verified, trusted training set. The decision criterion for self-verifying a training example is the prediction consensus of a previous snapshot of the network on multiple short sliding window excerpts of the training example at hand. On the unseen test data, an ensemble of three networks trained with this self-verification approach achieves a mean average precision (MAP@3) of 0.951. This is the second best out of 558 submissions to the corresponding Kaggle challenge.

Index Terms— Audio-tagging, Fully Convolutional Neural Networks, Noisy Labels, Label Self-Verification

1. INTRODUCTION

This short paper describes our approach[1] to the first "Freesound General-purpose Audio Tagging Challenge" which is carried out as Task 2 of the DCASE 2018 Challenge [1]. The central motivation for this challenge is to foster research towards more general machine listening systems capable of recognizing and discerning a wide range of acoustic events and audio scenes.

In particular, we aim at building an audio tagging system which assigns one out of 41 potential candidate labels to an unknown audio recording of arbitrary length. The labels comprise sound events such as music instruments, human sounds, animals, or domestic sounds [1]. What makes working with this data challenging is twofold: Firstly, the data set is collected from Freesound[2], which is a repository for user-generated audio recordings capturing diverse content with highly varying signal lengths recorded under diverse conditions [2, 3]. Secondly, the development (or training) dataset is delivered only partly with manually annotated ground truth labels. For the remaining recordings the labels are automatically generated and comprise up to 30-35% label noise in the "worst" categories. In the remainder of this paper, we refer to the manually annotated training observations as *verified* and to the additionally automatically annotated observations as *unverified*.

The central idea of our approach is to address the problem of noisy labels by training a fully convolutional audio classification network, which is iteratively fine-tuned by self-verifying the parts of the training observations provided without manually verified ground truth.

Technically, the task at hand is similar to other tasks (of earlier versions) of the DCASE challenge, which focus on detection or classification in audio scenes [4]. Therefore also our network architectures are inspired by earlier works addressing these problems [5]. Naturally, a large number of similar architectures and neural networks have proven to be powerful tools in this setting [6, 7, 8, 9, 10, 11]. There is also a large amount of prior work on training neural networks in the presence of label noise (see e.g. [12, 13, 14]). Due to the specifics of the current task (the labels of a large fraction of the data have verified manually and thus can be trusted), we opt for a straight forward iterative self-verification strategy. More concretely, to verify possibly noisy labels, our approach compares the labels of unverified examples with the predictions of a neural network, i.e. this can be interpreted as a version of "supervised" pseudo labeling [15].

2. AUDIO DATA PREPROCESSING

Before we present the data to our networks, we apply several audio preprocessing steps including silence clipping and spectrogram computation.

The first step in our pipeline is normalizing the audio signal to a dB-level of -0.1. Next, we clip potential silence in the beginning and the end of the normalized audio using *SoX*[3]. This step is important as we later on optimize our networks on short sliding windows of the original files and want to avoid presenting training observations containing only silence along with the original label of the file[4]. Figure 1 shows an audio example of class *Knock* where this preprocessing step has a severe impact, reducing the effective length of the spectrogram from 435 to only 186 frames. Note that the part of the spectrogram covering actual audio content is preserved.

Before computing the spectrograms we resample the audio signals to 32,000 Hz, and compute a Short Time Fourier Transform

[1] Code: `https://github.com/CPJKU/dcase_task2`
[2] `https://freesound.org/`
[3] `http://sox.sourceforge.net/`
[4] For further details on how the audio signals are preprocessed we refer to our write-up: `https://cpjku.github.io/dcase_task2/`

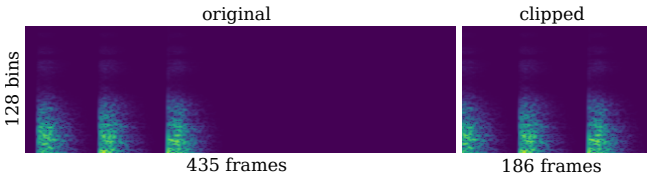


Figure 1: Log-Spectrogram (Version-2) of audio signal of class *Knock* before and after silence clipping.

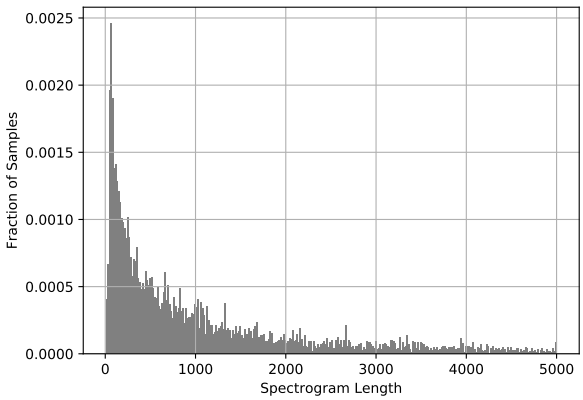


Figure 2: Distribution of spectrogram lengths computed with spectrogram *version-1*.

(STFT) using 1024-sample hann windows. To try to capture different aspects of the audio, we extract two different spectrogram versions for our final submission:

Version-1 uses an STFT hop-size of 192 samples. Given this spectrogram, we apply a perceptual weighting to the individual frequency bands of the power spectrogram [16][5]. Finally, we apply a mel-scaled filterbank yielding 128 frequency bins per data point.

Version-2 uses an STFT hop-size of 128 samples and does not apply perceptual weighting but takes the logarithm of the power spectrogram instead. Finally it is post-processed with a logarithmic filter-bank again resulting in 128 frequency bins [17].

An additional characteristic of the data at hand is the varying length of the recordings. Figure 2 shows the distribution of spectrogram lengths for spectrogram *version-1*. As convolutional neural networks – which are the central component of our method – in general have a fixed field of view (input dimensions), it is desirable to work with audio excerpts consisting of the same number of frames. Additionally, to design convolution networks including max-pooling layers with a certain depth, we need to exceed a minimum input dimensionality[6]. To that end, we fix a target length of 3000 frames and simply repeat a given excerpt in case it is too short and clip it at 3000 frames in case it is too long. The 3000 frame threshold is chosen intuitively as the spectrogram length distribution in Figure 2 has a long tail with few observations.

[5] `librosa.core.perceptual_weighting`

[6] After each max-pooling layer the dimensionality of the input / feature maps gets halved. This of course restricts the maximum depth of a network.

Input $1 \times 384 \times 128$
5×5 Conv(pad-2, stride-2)-64-BN-ReLU
3×3 Conv(pad-1, stride-1)-64-BN-ReLU
2×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-128-BN-ReLU
3×3 Conv(pad-1, stride-1)-128-BN-ReLU
2×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-256-BN-ReLU
Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-256-BN-ReLU
Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-384-BN-ReLU
Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-384-BN-ReLU
2×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-512-BN-ReLU
3×3 Conv(pad-1, stride-1)-512-BN-ReLU
1×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-1, stride-1)-512-BN-ReLU
3×3 Conv(pad-1, stride-1)-512-BN-ReLU
1×2 Max-Pooling + Drop-Out(0.3)
3×3 Conv(pad-0, stride-1)-512-BN-ReLU
Drop-Out(0.5)
1×1 Conv(pad-0, stride-1)-512-BN-ReLU
Drop-Out(0.5)
1×1 Conv(pad-0, stride-1)-41-BN
Global-Average-Pooling
41-way Soft-Max

Table 1: Network architecture. BN: Batch Normalization, ReLU: Rectified Linear Unit.

3. NETWORK AND TRAINING DETAILS

In this section, we describe the neural network architectures as well as the optimization strategies used for training our audio scene classification networks.

3.1. Network Architecture

Our basic network architecture is a fully convolutional neural network as depicted in Table 1. In total we use three slightly modified versions of this architecture for our submission, but the general design principles remain the same. The feature learning part of our model follows a *VGG style network* [18], and the classification part of the network is designed as a global average pooling layer [19] over 41 feature maps (one for each class) followed by a *softmax* activation. In our experiments, global average pooling over per-class feature maps consistently outperforms networks using fully connected layers as a classification output. As an activation function within the network we use Rectified Linear Units (ReLUs) in combination with batch normalization [20]. Overall, this model comprises 14,865,124 trainable parameters.

3.2. General Training Procedure

At training time, we show the network randomly selected 384 frame excerpts of the full spectrograms, while presenting the whole 3,000 frame spectrograms during testing. This is technically possible as

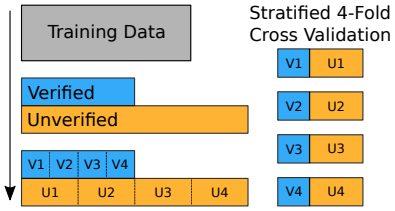


Figure 3: Stratified 4-fold cross-validation setup. After splitting into verified and unverified data we create eight stratified splits, each preserving the original label distribution. As a last step we assemble the eight sub-splits into four distinct folds.

our network is fully convolutional and can therefore process audio excerpts of varying length. Intuitively, presenting shorter excerpts for training should mitigate the effect of over-fitting to the individual training examples, as we end up with a much larger amount of shorter sub-excerpts. To further prevent over-fitting, we additionally apply mixup-data augmentation [21] with an $alpha$ of 0.3.

As optimizer we use the ADAM update rule [22] with an initial learning rate of 0.001 and a mini-batch size of 100 samples for training the initial version of our models. Each model is trained for 500 epochs, where we linearly decay the learning rate to zero starting from epoch 100. When fine tuning our models with iterative self-verification (see Section 4), we use a slightly modified training setup as described below.

4. 4-FOLD ITERATIVE SELF-VERIFICATION

This section describes our iterative self-verification loop to address the noisy labels in the development dataset. The central idea is to gradually shift unverified labels into the verified, trusted training set for fine-tuning the models.

4.1. 4-Fold Cross-Validation Setup

One crucial component of our self-verification approach is the way we prepare our training-validation-setup. To enable the proposed step-wise verification approach, we have to design our folds in a way that parts of the data (which we would like to verify) are never presented to the verification model for training. Otherwise, the neural network would learn the entire training set by heart, even in the presence of noisy labels [23], and its prediction become worthless for the verification strategy. Figure 3 provides an overview on how we prepare our split. First, we separate the development dataset into verified and unverified observations. Second, we split each of the two subsets into four stratified sub-folds, meaning that the label distribution in the sub-folds remains the same as in the original dataset. We consider this an important detail as the challenge organizers state on the official web page that the unseen test data exhibits a similar label distribution. In fact, for selecting our final submission we did not rely on the public Kaggle-Leaderboard but on the average performance on our local 4-fold cross-validation setup. Each of our local validation folds contains approximately 900 verified files which is substantially more than the 300 files (19% of official test data) considered for the public leaderboard. Although preparing this validation setup is straight forward, it is a crucial step as it is the basis for our self-verification pipeline described below.

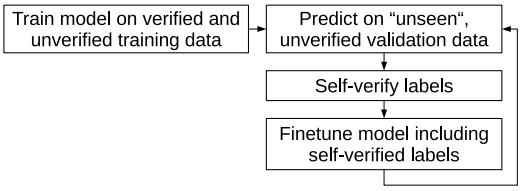


Figure 4: Iterative self-verification loop and model fine-tuning.

4.2. Iterative Self-Verification Loop

Figure 4 provides an overview of our iterative self-verification loop. Step one of this procedure is to train an initial model utilizing all the training data of a fold also including the unverified samples. We train one model for each fold as described in Section 3.2 and keep the best parametrization according to its validation loss on the verified validation data excluded from training. Note that the unverified data is not considered for model selection. This is also the main reason why we provide separate stratified splits for verified and unverified observations as it should provide us with an as reliable estimate of the real model performance as possible.

Once the initial model is trained we use it to predict the labels of its respective unseen validation examples. However, in contrast to the model selection we now only predict on the unverified observations. In particular, we draw K random 384 frame excerpts of the original 3000 frame spectrograms of the audio clip to verify. We then compute the average of the individual posterior class distributions $p_i(y|x)$ of these K excerpts:

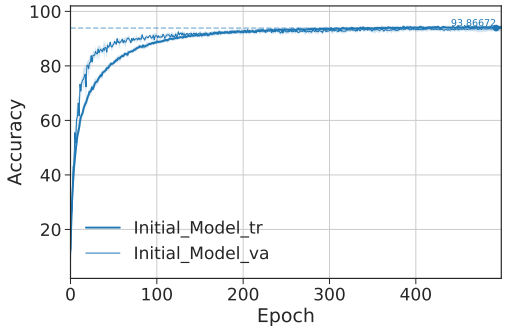
$$\bar{p}(y|x) = \frac{1}{K}\sum_{i=1}^{K} p_i(y|x) \tag{1}$$

with $K = 25$, $x \in \mathbb{R}^{384\times128}$ and $y \in \{0, ..., 40\}$. We then proceed by considering unverified examples as correctly annotated if

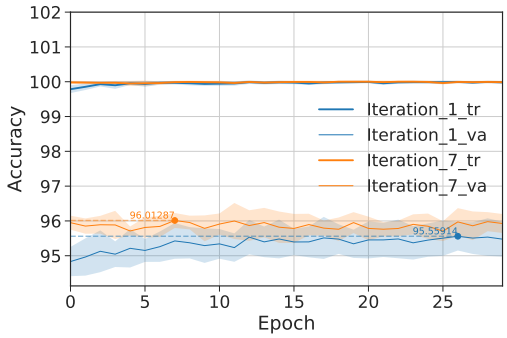
1. The provided unverified label y_u matches the label $\underset{i}{\text{argmax}}\ \bar{p}(y|x)$ predicted by the average of the individual posterior distributions.
2. The average of the target class posteriors exceeds 0.95.
3. A maximum count of 40 self-verified examples per class is not yet reached.

The intuition behind this approach is that especially for the unverified training examples multiple different classes might be present in a single audio recording. Still it is annotated with a single and hence unreliable label. When predicting on multiple random sub-excerpts of the recording this should be revealed by exhibiting a low average posterior probability $\bar{p}(y|x)$ for the provided target class label. The last condition for self-verification is introduced as some classes have very few examples and we want to avoid shifting the original label distribution of the dataset. As our procedure is based on four distinct cross-validation folds each unverified example is considered for verification once per iteration.

The final stage of this self-verification loop is to fine-tune the four initial fold models, this time using the officially provided verified observations and the ones passing the self-verification conditions in the previous stage. For the fine-tuning stage we train for 30 epochs with an initial learning rate of 0.0002, which is again decayed to zero starting after five epochs. We do not use mixup



(a) Initial model trained on entire dataset.



(b) Models after self-verification fine-tuning.

Figure 5: Comparison of model performance of network trained on (a) all provided data including noisy labels and (b) all data which passed the proposed self-verification at the respective iteration. We report mean and standard deviation (shaded area) of the classification accuracy on training (tr) and validation (va) set averaged across our four development folds. For the present case self-verification iteration 7 yields the best model (fine-tuned for 7 $\times$ 30 epochs).

data augmentation in this stage anymore. After fine-tuning we go back to step two and repeat the whole procedure in a loop for ten times. Once the ten iterations are completed, we select the model parameterization of the iteration showing the lowest average validation loss on the officially verified validation data. Note that we do not reset the network to the initial parameterization after each fine-tuning iteration but continue training the same model.

5. EXPERIMENTAL RESULTS

In this section we report our empirical results on both, our local validation setup as well as the public and private Kaggle-Leaderboard. As evaluation measures we consider the Mean Average Precision @ 3 (MAP@3), which is the score officially used for the challenge, as well as classification accuracy and F-Score when presenting the performance on individual classes.

Figure 5a shows training and validation accuracy on our local 4-fold setup of the model described in Table 1 when training on the entire dataset including unverified labels. This corresponds to the first stage of the self-verification procedure in Figure 4. We report the mean and standard deviation of the network across the four distinct folds where the best average accuracy on the validation set after stage one is 93.87%. In Figure 5b we show the performance of the same model after the first and seventh fine-tuning iteration.

	Public	Private	Public & Private
MAP@3	0.9563	0.9518	0.9526
Accuracy	93.688	92.610	92.812

Table 2: Audio tagging performance on the *public*, *private*, and *public & private* test set (Kaggle-Leaderboard).

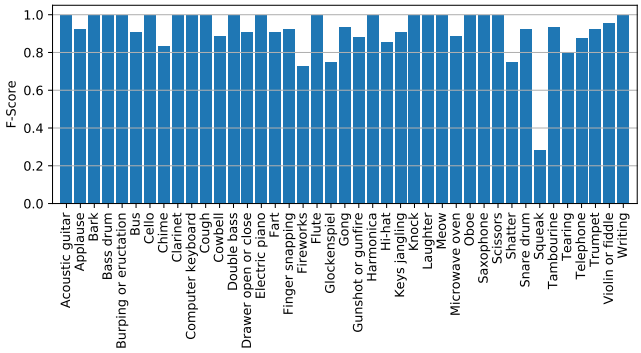


Figure 6: F-score of final submission on individual classes.

Already after the first self-verification iteration, we observe a performance improvement to 95.56%. Finally, the model reaches its best performance in iteration seven with an average verified validation set performance of 96.01%.

Before preparing our submission for the challenge, we trained three similar networks to the one in Table 1 (two on spectrogram *version-1* and one on spectrogram *version-2*) and averaged their predictions. When evaluating this submission on the official test data set we achieve the second best scoring submission in the final private Kaggle-Leaderboard with a MAP@3 of 0.9518. For easier comparability with future research on this dataset, we also report detailed results on the different subsets of the test data in Table 2.

To provide an intuition on how well the approach performs on individual classes, we report the F-score for all 41 classes in Figure 6. We observe that the model achieves an F-score above 0.8 for all classes except for *Squeak*, *Fireworks*, and *Glockenspiel*. Many of the classes are even recognized with a perfect score of 1.0. Considering the noisy labels and that there are 41 different classes to distinguish we take this as a remarkable result.

6. CONCLUSION

In this workshop paper, we described our submission to the first Freesound general-purpose audio tagging challenge carried out with DCASE 2018. Our proposed approach is an iterative self-verification loop built on top of a fully convolutional neural network. After an initial training stage using all the data, we iteratively fine-tune our networks using label self-verification. The central idea of our proposal is to add unverified examples step-by-step to the training set based on the prediction consensus of our networks with the suggested, noisy labels. For a single model, this approach improves the classification accuracy from 93.87% to 96.01% on the local validation set. When training an ensemble of three similar networks in this fashion, we are able to achieve a MAP@3 of 0.9518 (92.610% accuracy) on the final private Kaggle-Leaderboard. Overall this yields the second best scoring out of 558 submissions.

7. REFERENCES

[1] 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," 2018, submitted to DCASE2018 Workshop. [Online]. Available: https://arxiv.org/abs/1807.09902

[2] 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 *Proceedings of the 18th International Society for Music Information (ISMIR)*, Suzhou, China, 2017, pp. 486–493.

[3] J. F. Gemmeke, D. P. W. 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 *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, USA, 2017, pp. 776–780.

[4] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," 2018, submitted to DCASE2018 Workshop. [Online]. Available: https://arxiv.org/abs/1807.09840

[5] H. Eghbal-Zadeh, B. Lehner, M. Dorfer, and G. Widmer, "Cp-jku submissions for dcase-2016: A hybrid approach using binaural i-vectors and deep convolutional neural networks," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

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

[7] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *International Workshop on Machine Learning for Signal Processing (MLSP)*, Boston, USA, 2015, pp. 1–6.

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

[9] T. H. Vu and J.-C. Wang, "Acoustic scene and event recognition using recurrent neural networks," *Detection and Classification of Acoustic Scenes and Events*, vol. 2016, 2016.

[10] K. Choi, G. Fazekas, and M. B. Sandler, "Automatic tagging using deep convolutional neural networks," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York City, United States, 2016, pp. 805–811.

[11] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley, "Convolutional gated recurrent neural network incorporating spatial features for audio tagging," in *2017 International Joint Conference on Neural Networks (IJCNN)*, Anchorage, USA, 2017, pp. 3461–3466.

[12] S. Sukhbaatar and R. Fergus, "Learning from noisy labels with deep neural networks," *CoRR*, vol. abs/1406.2080, 2014.

[13] S. E. Reed, H. Lee, D. Anguelov, C. Szegedy, D. Erhan, and A. Rabinovich, "Training deep neural networks on noisy labels with bootstrapping," *CoRR*, vol. abs/1412.6596, 2014.

[14] L. Jiang, Z. Zhou, T. Leung, L. Li, and L. Fei-Fei, "Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, Stockholm, Sweden, 2018, pp. 2309–2318.

[15] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on Challenges in Representation Learning, ICML*, vol. 3, 2013, p. 2.

[16] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.

[17] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "Madmom: A new python audio and music signal processing library," in *Proceedings of the 2016 ACM on Multimedia Conference*, Amsterdam, The Netherlands, 2016, pp. 1174–1178.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, USA, 2015.

[19] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proceedings of the International Conference on Learning Representations (ICLR)*, Banff, Canada, 2014.

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, Lille, France, 2015, pp. 448–456.

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

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, USA, 2015.

[23] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *In Proceedings of the 5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2016.