ENSEMBLE OF CONVOLUTIONAL NEURAL NETWORKS FOR GENERAL-PURPOSE AUDIO TAGGING

Bogdan Pantic

School of Electrical Engineering
Signals and Systems Department
Belgrade, Serbia
bogdan.pantic@yahoo.com

ABSTRACT

This work describes our solution for the general-purpose audio tagging task of DCASE 2018 challenge. We propose the ensemble of several Convolutional Neural Networks (CNNs) with different properties. Logistic regression is used as a meta-classifier to produce final predictions. Experiments demonstrate that the ensemble outperforms each CNN individually. Finally, the proposed system achieves Mean Average Precision (MAP) score of 0.945 on test set, which is a significant improvement compared to the baseline.

***Index Terms*—** audio tagging, DCASE 2018, convolutional neural networks, ensembling

1. INTRODUCTION

The goal of audio tagging is to create models capable of recognizing a variety of sounds. Those include musical instruments, vehicles, animals, sounds generated by some sort of human activity etc. The motivation for a research in the field of an artificial sound understanding can be found in potential applications such as security, healthcare (hearing impairment), improvements in smart devices, various music related tasks etc. Detection and Classification of Acoustic Scenes and Events (DCASE) challenge 2018 consists of several tasks which provide the way to evaluate different methods for solving problems related to non-speech audio signals. The focus of this paper will be on the task 2: "General-purpose audio tagging of FreeSound content with AudioSet labels" which is hosted on Kaggle platform [1]. The dataset contains around 9500 training and 1600 testing examples which belong to one of 41 unequally distributed classes (bus, gunshot, knock, flute, etc.). Audio files differ in length with the duration ranging from 300ms to 30s. All samples were automatically annotated, but only a portion of training set labels were manually verified. Therefore, there is a large variation in label quality which poses yet another problem to participants – to extract as much information as possible from the weakly labeled data. Another major issue is a label density. It represents a portion of audio in which the tagged event is actually present. As one can imagine, the label density can vary significantly, so creating models which can successfully tackle that is of high importance.

Though the research in this area has recently expanded, a related work can be found at the previous editions of DCASE challenge. Alternatively, a related research can also be found in the area of Music Information Retrieval (MIR). The earlier research mostly relied on hand crafted features and shallow models. For example, in the first edition of DCASE in 2013 models like SVM [2] and bagging of decision trees [3] were used with the variety of features. Similar tendency can be found in the MIR research where features were particularly designed to capture timbral and rhythmic characteristics [4]. The later research shows an obvious shift towards feature learning, more precisely, deep learning. Following their success in computer vision, convolutional neural networks (CNN) are extensively used for the audio scene classification [5, 6], event detection [7, 8], music tagging [9] etc. One can use CNNs in different settings and on different input representations. Using raw audio with one-dimensional convolutions is a viable option, but most research relies on some sort of time-frequency representation and two-dimensional CNNs as it is typically expressive enough and less computationally expensive. Mel-spectrograms are widely used, but Constant Q transform (CQT) also shows promising results [10]. Although a computer vision inspired the rapid growth of CNN usage, the interpretation in audio domain fundamentally differs. While vertical and horizontal axes in images generally satisfy the same properties and should be treated equivalently, time and frequency axes of an audio signal represent different modalities. Therefore, there is a room for the domain specific filter design, which should capture interesting patterns, improve CNN architecture efficiency and hopefully increase model performance. Several researchers have already tried to exploit these facts, yielding competitive results in related areas. Depending on a problem in question, approaches focus on modelling temporal [11] and frequency [12] related features with horizontal and vertical filters respectively. Especially interesting for our dataset are wide architectures that incorporate parallel feature learning [13, 14]. In that setting, one should be able to use many different filter shapes and fuse extracted features in later layers, which would enable the model to learn much richer set of descriptors. Due to the nature of our problem, mainly, large differences in acoustic properties of provided classes, parallel architectures could prove to be beneficial.

This paper describes our solution for DCASE challenge. We will evaluate several architectures and preprocessing techniques. The main goal is to design diverse set of classifiers and leverage these differences by stacking predictions of individual models. The paper is organized as follows. Validation, preprocessing, proposed models and ensembling are described in the section 2. Section 3 deals with the evaluation and details of experimental setup. Finally, the obtained results are presented in the section 4.

2. SYSTEM ARCHITECTURE

This section provides an overview of the most crucial aspects of the solution, including validation setup, data preprocessing, CNN

architectures and ensembling technique.

2.1. Validation Setup

One of the major decisions during the development of the machine learning system is a configuration of the train-validation split. It is common to use K-fold cross-validation, where a model is trained on K-1 folds and validated on the remaining one. At the end, the average score is used as a performance estimate. However, in the case of the provided dataset, there is a large percent of samples which are not manually verified in the training set and none of them in the test set. Since validation should represent unseen data as close as possible, we choose to use only manually verified examples. The same split is used for all models, where 10% of the data is used for validation. It is also worth mentioning that train and validation data have the same distribution of labels, but the distribution of manually verified samples of different classes in the training set is not uniform.

2.2. Preprocessing

In the introduction, it is pointed out that audio files have different length and consequentially don't contain the same amount of information. Therefore, the preprocessing should account for both fixed size input requirement of our models and the information perseverance. The input audio length is predefined, and it varies between 4 and 8 seconds for different models while shorter files are zero padded. The sampling rate is 44.1 kHz. Two representations are used: mel-spectrogram with 96 mel bands and constant Q transform with 96 or 110 bands. Longer files are split into chunks of predefined length with several overlap values and the resulting spectrograms are converted to dB-scale (the amplitude is scaled relative to maximum value). Obtained inputs are of shape (n_{bands}, n_{frames}) and they are all used as new training examples (which means that the resulting dataset is larger than the initial one). Finally, the data is standardized (by subtracting the mean and dividing by the standard deviation of the entire training set). At the test time, identical transformations are applied and the results are generated as an average of the predictions of each chunk corresponding to the same file. The entire preprocessing is done using Librosa library [15], and the remaining hyper-parameters are left to the default values.

2.3. Network Architectures

The ensembling is known to yield the highest benefits when predictions of the base models are less correlated. To fully exploit that fact, we propose a couple of architectures with slightly different properties.

The first network is inspired by one of the top solutions of "Tensorflow Speech Recognition challenge" [16] and suggested by other participant of DCASE challenge [17]. The initial convolutional layer has 64 filters of shape 7x3, followed by 4x1 max-pooling layer. The next layer contains 128 filters of shape 7x1 and 4x2 max-pooling with 2x2 strides. Finally, two convolutional layers with 128 filters and 1x5 and 5x1 shapes respectively are stacked before the global-max-pooling layer. Two densely-connected layers with 64 neurons are used for an additional feature extraction before a softmax classifier. The activation function of each layer is rectified linear unit and each convolutional layer is followed by batch-normalization. Dropout of 0.25 is used before each dense layer for additional regularization.

The second architecture is proposed in [13]. It relies heavily on the domain knowledge, by introducing sets of rectangular filters applied in parallel on input. On the one hand, for frequency related features, vertical filters which cover 90% and 40% of domain are used with small temporal dimension. On the other hand, to capture temporal features efficiently, an average pooling is applied over frequency axis of spectrogram and several 1D convolutional kernels are employed. The filter lengths are 165, 128, 64 and 32. Outputs of both frequency and time related feature extractors are concatenated. Three 2D convolutional layers with 512 filters are then applied, the result is flattened and the dense layer with 300 neurons followed by 0.4 dropout is added before the output layer.

Additionally, to explore other aspects of rectangular filter design, a new architecture is proposed. It is inspired by [12] and the details are given in the table 1:

Table 1: Description of model 3: Set of rectangular filters, concatenation of feature maps and additional layers

Conv1: 48x (8x7)| 32x (32x7)| 16x (64x7)| 16x (90x7) + BN
Concatenate
Max-Pooling (5x5)
Conv2: 120x (2x2)
Global-Max-Pooling 2D
Dense1: 64 units + Dropout 0.2
Dense2: 64 units + Dropout 0.2
Dense3: 41 units + softmax

The output of each branch in *Conv1* layer has to be the same, so that feature maps can be stacked. This is achieved by zero-padding input accordingly. All hidden layers use rectified linear unit as activation.

The combination of previously discussed ideas has led to yet another model:

Table 2: Description of model 4: Set of rectangular filters with more depth, concatenation of feature maps, additional convolutional and dense layers

Conv1: 64 x (8x3)| 64 x (16x3)| 64 x (32x3)
Max1:Max-Pooling (4x1) + BN
Conv2: 128 x (8x1)| 128 x (16x1)| 128 x (32x1)
Max2: Max-Pooling (4x2) + BN
Concatenate
Conv3: 128 x (5x1) + BN
Conv4: 128 x (1x5) + BN
Global-Max-Pooling 2D
Dense1: 64 units + Dropout 0.2
Dense2: 64 units + Dropout 0.2
Dense3: 41 units + softmax

The fourth model is using architectural designs of the first network, but with the parallelism introduced in the models two and three. Instead of the single convolutional layer before the concatenation, it is using two layers per branch. Same padding is used in these two layers to avoid a dimensionality mismatch. The strides of max-pooling layers are 2x1 and 2x2 respectively. Similarly, hidden layers use ReLU activation.

Models which are typically used in a computer vision community can be added to maximize diversity. Concretely, we use

Inception V3 [18] and MobileNet [19] with weights pretrained on ImageNet. They are implemented using Keras [20] library. The classification layer is removed from both architectures and two layers with 64 units and 0.2 dropout are added before the softmax layer with 41 units. The additional preprocessing steps are required for these setups. We had to resize the inputs to 150x150 for Inception and 160x160 for MobileNet to match implementation requirements. Also, a number of channels had to be matched, so mean is calculated across the entire training data and added as the second and the third channel to each sample. These models require more computing time, but add a significant value to ensembles. We will refer to Inception as model 5 and MobileNet as model 6 in the remainder of the paper.

2.4. Ensembling

Once models are configured and trained there are many ways to leverage generated predictions. Calculating arithmetic or geometric mean are two obvious ways, since they don't add additional complexity and almost always improve performance. However, once we have a sufficiently diverse set of base predictions, real gains come with stacking. Stacking is performed by using predictions as features for a meta-model. It is often done in a cross-validation setting, but because of the train-validation split used by level-1 models, we are constrained to use only 10% of data for meta-model training. The predictions of the individual classifiers are stacked in the columns for both the validation and the test set. For example, each of ten models would have 41 (number of classes) predictions per example and the resulting feature matrix would have 410 columns. The validation data then becomes a new training set and stratified 5-fold cross-validation is used for training. The experiments have shown that logistic regression is suitable candidate for the meta-classifier. Principal component analysis (PCA) is used for a dimensionality reduction in each of K iterations. The predictions of the meta-model on the validation data are combined and MAP score is computed to produce a new performance estimate. Finally, trained meta-model is used to generate test set predictions.

3. EXPERIMENTAL DESIGN

3.1. Evaluation

Organizers split test data in a public and a private part. Participants submit their predictions for the entire test set, but they can only see a public score (contains around 19% of test data). Submissions are evaluated using mean average precision:

$$MAP = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{min(n,3)} P(j) \quad (1)$$

where N is a number of audio files used for scoring, n is a number of predictions per file and $P(j)$ is the precision at cutoff j. The private score is released after the competition ends.

3.2. Hyper-parameters and data augmentation

The input is split into patches of length 4s, 5s or 6s with 1s overlap, or 8s with 2s overlap. The experiments on shorter inputs gave worse scores and didn't add any value to the ensembles, so they were discarded. The networks were trained using categorical cross-entropy as a loss function. Adam is used as an optimizer, with the initial learning rate of 0.001. The mini-batch size was 32 or 64, depending on a model and the input size. During training, we monitor a validation loss and save currently the best performing model. If the validation loss doesn't decrease for seven epochs, the learning rate is multiplied by 0.5. Early-stopping is used to avoid overfitting. The training stops after 20 epochs passed from the last improvement. A maximum number of epochs for all models is 250.

Data augmentation is another way to reduce overfitting. Transformations are applied to the original data points, artificially enlarging dataset. It's crucial that augmentation techniques do not change a true label of the particular sample, otherwise performance may decrease. Concretely, we used random width shift and zoom with maximum range of 0.1. Another interesting augmentation technique which significantly reduced overfitting is random erasing [21]. It works by randomly selecting rectangular area on the input image and changing its values with random numbers. Finally, the most important augmentation technique used is mixup [22]. It is implemented by creating virtual feature-target pairs $(\tilde{x}, \tilde{y})$:

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j \quad (2)$$
$$\tilde{y} = \lambda y_i + (1-\lambda)y_j \quad (3)$$

where (x_i, y_i) and (x_j, y_j) are the pairs drawn randomly (regardless of the provided label of the sample) from the training data, while $\lambda \sim Beta(\alpha, \alpha)$. Therefore, the parameter α affects a regularization strength (larger α implies stronger regularization). Mixup is encouraging linear behavior between training examples which has other positive side effects. The original paper shows that it also improves robustness to corrupted labels. They argue that by increasing α it should be possible to create virtual examples further from the original, wrongly labeled samples and therefore reduce effect of the memorization of the corrupted classes. As discussed previously, the majority of the training examples were not manually verified (accuracy of those labels is estimated to be at least 65-70% per class), so mixup allows us to decrease negative impact of the label noise with minimal additional computational requirements, since each "new" training example is just a linear combination of two original samples. These desirable properties have made mixup a crucial part of every pipeline. Regarding the parameter value, α between 0.2 and 0.3 was found to be optimal across different architectures. Every augmentation technique is performed during the training phase.

4. RESULTS

In this section, the results of the proposed architectures are presented and discussed. Table 3 summarizes important information regarding the models and their scores on a test set (public and private part combined). For clarity, models are specified only through ids, with respect to the order of presentation. These particular combinations of the models and the preprocessing techniques have been chosen based on the estimated performance of the ensemble on the validation data. Additionally, to avoid the usage of too many configurations in the final ensemble, the models

which have brought only small improvements haven't been included. We can come up with several interesting conclusions by inspecting these values. The experiments have shown that CQT outperforms mel-spectrogram for most models. Nevertheless, mel-spectrograms have added significant diversity, so they have been kept for the stacked ensemble. Also, the inputs of length 4s and 5s seem to be optimal, since they provide nice trade-off between performance, required computation and the memory usage. The larger number of bands certainly helps, but it also increases computation time.

Organizers provided a baseline model for comparison with the proposed solutions. Its inputs are log scale mel-spectrograms with windows of length 0.25s and hop size of 0.125s. The model contains 3 convolutional layers with filter shapes 7x7, 5x5 and 3x3 respectively before the output softmax layer. Total number of parameters is around 658.1K. It achieves MAP score of approximately 0.70 on test set (0.7 on public and 0.69 on private leaderboard).

Table 3: Summary of architectures: Models, length of audio chunks, overlap used for longer files, input representations, MAP scores

Id	Length	Overlap	Transform	Bands	MAP
1	4s	1s	Mel-spec	96	0.87
1	5s	1s	CQT	110	0.913
1	6s	1s	Mel-spec	96	0.878
1	8s	2s	CQT	110	0.909
2	4s	1s	CQT	96	0.915
2	5s	1s	Mel-spec	96	0.889
3	4s	1s	CQT	96	0.872
4	4s	1s	CQT	96	0.908
5	4s	1s	CQT	96	0.895
5	5s	1s	CQT	96	0.894
6	4s	1s	CQT	110	0.909

Final solution is an ensemble of 11 proposed configurations. The meta-model is logistic regression with the regularization parameter $C = 4$. It is trained on 120 features, after PCA dimensionality reduction. The average precision scores per class (taking into account only top three predictions per example, referred to as AP@3) are shown in Table 4. Considering high overall performance of both low level classifiers and the ensemble, the results are expected, since most of the classes obtained nearly perfect scores. However, there are a few exceptions, most notably: "Squeak", "Fireworks" and "Scissors". "Squeak" is a class which has one of the lowest percentages of the manually verified examples, which could be a reason for the bad score. "Fireworks" are, intuitively, often confused with gunshots, which seems natural and it is something that would be problematic even for a human listener. Finally, the class "Scissors" has the 2nd smallest number of samples in the training set and the smallest in the test set. There are a couple of less problematic classes like "Chime" and "Glockenspiel" (often confused), "Gunshot, gunfire" (same as fireworks), "Bus" and "Telephone".

The proposed ensemble achieves MAP of 0.956 on the public leaderboard, 0.942 on the private leaderboard and 0.945 on the entire test set, which is an improvement over level-1 models and the baseline. The final submission ranked 12th among 558 competing teams on the private leaderboard.

5. CONCLUSION

This paper proposes an ensemble of convolutional neural networks for the classification of general audio signals. We have introduced several architectures, preprocessing techniques and validation setup in order to get a diverse set of base predictions. Logistic regression is then used as a meta-model to obtain the final output. It has been shown that it outperforms individual models substantially, which demonstrates that original architectures really provide sufficiently diverse information. Further improvements might be possible by including different non-deep learning models with hand crafted features, additional data augmentation or by adding more pre-trained models to the ensemble.

Table 4: Per-category results: Class, number of samples and AP@3

Class	Samples	AP@3
Acoustic guitar	45	0.93
Applause	32	1.0
Bark	28	0.964
Bass drum	28	1.0
Burping, eructation	32	1.0
Bus	25	0.873
Cello	54	0.981
Chime	29	0.896
Clarinet	56	0.988
Computer keyboard	26	0.904
Cough	30	1.0
Cowbell	42	1.0
Double bass	40	0.987
Drawer open, close	29	0.931
Electric piano	32	0.969
Fart	30	0.983
Finger snapping	33	1.0
Fireworks	32	0.76
Flute	55	0.972
Glockenspiel	29	0.868
Gong	37	1.0
Gunshot, gunfire	63	0.889
Harmonica	33	0.955
Hi-hat	39	0.949
Keys jangling	28	0.946
Knock	39	0.957
Laughter	38	0.947
Meow	29	1.0
Microwave oven	29	0.983
Oboe	42	0.96
Saxophone	110	0.958
Scissors	25	0.78
Shatter	29	0.983
Snare drum	34	0.917
Squeak	29	0.672
Tambourine	40	0.95
Tearing	27	0.962
Telephone	48	0.809
Trumpet	37	0.946
Violin, fiddle	108	0.995
Writing	29	0.943

6. REFERENCES

[1] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, X. Favory, J. Pons, X. Serra, "General-purpose Tagging of Freesound Audio with AudioSet Labels: Task Description, Dataset, and Baseline," Submitted to *DCASE 2018 workshop*, 2018.

[2] J.T. Geiger, B. Schuller, G. Rigoll, "Recognizing acoustic features with large-scale audio feature extraction and SVM," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.

[3] D. Li, J. Tam, D. Toub, "Auditory scene classification using machine learning techniques," *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events*, 2013.

[4] G. Tzanetakis, P. Cook, "Musical genre classification of audio signals*," IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, July 2002.

[5] M. Valenti, A. Diment, G. Parascandolo, S. Squartini, T. Virtanen, "DCASE 2016 acoustic scene classification using convolutional neural networks," *Detection and Classification of Acoustic Scenes and Events*, 2016.

[6] S. Park, S. Mun, Y. Lee, H. Ko, "Acoustic scene classification based on convolutional neural network using double image features," *Detection and Classification of Acoustic Scenes and Events*, 2017.

[7] D. Lee, S. Lee, Y. Han, K. Lee, "Ensemble of convolutional neural networks for weakly supervised sound event detection using multiple scale input," *Detection and Classification of Acoustic Scenes and Events*, 2017.

[8] I. Jeong, S. Lee, Y. Han, K.Lee, "Audio event detection using multiple-input convolutional neural network," *Detection and Classification of Acoustic Scenes and Events*, 2017.

[9] K. Choi, G. Fazekas, M. Sandler, "Automatic tagging using deep convolutional neural networks," *International Society of Music Information Retrieval Conference*, 2016.

[10] T. Lidy, A. Schindler, "CQT-based convolutional neural networks for audio scene classification and domestic audio tagging," *Detection and Classification of Acoustic Scenes and Events*, 2016.

[11] J. Schluter, S. Bock, "Improved musical onset detection with convolutional neural networks," *IEEE International Conference on Acoustic, Speech and Signal Processing*, 2014.

[12] E. Fonseca, R. Gong, D. Bogdanov, O. Slizovskaia, E. Gomez, X. Serra, "Acoustic scene classification by ensembling gradient boosting machine and convolutional neural networks," *Detection and Classification of Acoustic Scenes and Events*, 2017.

[13] J. Pons, O. Nieto, M. Prockup, E. Schmidt, A. Ehmann, X. Serra, "End-to-end learning for music audio tagging at scale," in *Proceedings of International Society for Music Information Retrieval Conference*, 2018.

[14] A. Schindler, T. Lidy, A. Rauber, "Multi-temporal resolution convolutional neural networks for the DCASE acoustic scene classification task," *Detection and Classification of Acoustic Scenes and Events*, 2017.

[15] https://librosa.github.io/librosa/

[16] https://www.kaggle.com/c/tensorflow-speech-recognition-challenge/discussion/47715

[17] https://www.kaggle.com/c/freesound-audio-tagging/discussion/57051

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, "Rethinking the inception architecture for computer vision," *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv: 1704.04861*, 2017.

[20] https://keras.io/

[21] Z. Zhong, L. Zheng, G. Kang, S. Li, Y. Yang, "Random erasing data augmentation," *arXiv preprint arXiv: 1708.04896*, 2017.

[22] H. Zhang, M. Cisse, Y.N. Dauphin, D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," *International Conference on Learning Representations*, 2018.