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 the DCASE 2018 challenge. We propose ensemble of several Convolutional Neural Networks (CNNs) with different properties. Logistic regression is used as meta-classifier to produce final predictions. Experiments demonstrate that ensemble outperforms each CNN individually. Finally, proposed system achieves Mean Average Precision (MAP) score of 0.956 on public leaderboard, which is significant improvement compared to baseline.

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

1. INTRODUCTION

The goal of general audio tagging is to create models capable of recognizing variety of sounds. Those include musical instruments, vehicles, animals, sounds generated by some sort of human activity etc. Motivation for research in the field of 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 way to evaluate different methods for solving problems related to non-speech audio signals. Focus of this paper will be on task 2: "General purpose audio tagging of FreeSound content with AudioSet labels" which is hosted on Kaggle platform [1]. 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 duration ranging from 300ms to 30s. All samples were automatically annotated, but only portion of training set labels were manually verified. Therefore, there is large variation in label quality which poses yet another problem to participants – to extract as much information possible from weakly labeled data. Another major issue is label density. It represents portion of audio in which tagged event is actually present. As one can imagine, label density can vary significantly, so creating models which can successfully tackle it is of high importance.

Though research in this area has recently expanded, related work can be found at previous editions of DCASE challenge. Alternatively, related research can also be found in the area of Music Information Retrieval (MIR). 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 variety of features. Similar tendency can be found in MIR research where features were particularly designed to capture timbral and rhythmic characteristics [4]. Later research shows obvious shift towards feature learning, more precisely, deep learning. Following their success in computer vision, convolutional neural networks (CNN) are extensively used for 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 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. Widely used are mel-spectrograms, but Constant Q transform (CQT) also shows promising results [10]. Although computer vision inspired rapid growth of CNN usage, interpretation in audio domain fundamentally differs. While vertical and horizontal axes in images generally satisfy same properties and should be threated equivalently, time and frequency axes of audio signal represent different modalities. Therefore, there is room for domain specific filter design, which should capture interesting patterns, improve CNN architecture efficiency and hopefully increase model performance. Several researchers already tried to exploit these facts, yielding competitive results in related areas. Depending on 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 them in later layers, which would enable model to learn much richer set of descriptors. Due to 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. Main goal is to design diverse set of classifiers and leverage those differences by stacking predictions of individual models. Paper is organized as follows. Validation, preprocessing and proposed models are described in section 2. Section 3 deals with evaluation and details of experimental setup. Finally, obtained results are presented in section 4.

2. SYSTEM ARCHITECTURE

This chapter gives overview of crucial aspects of proposed solution, including validation setup, data preprocessing and CNN architectures.

2.1. Validation Setup

One of the major decisions during machine learning system development is configuration of train-validation split. It is common to use K-fold cross-validation, where model is trained on K-1 folds and validated on remaining one. At the end, average score is used as performance estimate. However, in the case of provided dataset, there is 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. Same split is used for all models, where 10% of data is used for validation. It is also worth mentioning that train and validation data have same distribution of labels, but distribution of manually verified samples of different classes in 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 same amount of information. Therefore, preprocessing should account for both fixed size input requirement of our models and information perseverance. Input audio length is predefined, and it varies between 4 and 8 seconds for different models while shorter files are zero padded. Two representations are used: mel-spectrogram with 96 mel bands and constant Q transform with 96 or 110 bins. Longer files are split into chunks of predefined length with several overlap values and resulting spectrograms are converted to dB-scale (amplitude is scaled relative to maximum value). Finally, obtained data is standardized. At test time, identical transformations are applied and results are generated as average of predictions of each chunk corresponding to the same file. Entire preprocessing was done using Librosa library [15].

2.3. Network Architectures

Ensembling is known to yield the highest benefits when predictions of base models are less correlated. To fully exploit that fact, we propose 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]. Initial convolutional layer has 64 filters of shape 7x3, followed by 4x1 max-pooling layer. 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 global-max-pooling layer. Two densely-connected layers with 64 neurons are used for additional feature extraction before softmax classifier. Activation function of each layer is rectified linear unit, while every 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 domain knowledge, by introducing sets of rectangular filters applied in parallel on input. 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, average pooling is applied over frequency axis of spectrogram and several 1D convolutional kernels are applied. 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, result is flattened and dense layer with 300 neurons followed by 0.4 dropout is added before output layer.

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

Table 1: Model 3

Conv1: 48x (8x7)| 32x (32x 7)| 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

Output of each branch in *Conv1* layer has to be 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.

Combination of previously discussed ideas led to yet another model:

Table 2: Model 4

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

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

3. EXPERIMENTAL DESIGN

3.1. Evaluation

Organizers split test data in public and private part. Participants submit their predictions for entire test set, but they can only see 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 number of audio files used for scoring, n is number of predictions per file and $P(j)$ is precision at cutoff j. Private score is released after competition ends.

3.2. Hyper-parameters and data augmentation

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

Data augmentation is another way to reduce overfitting. Transformations are applied on original data points, artificially enlarging dataset. It's crucial that augmentation techniques do not change true label of 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 input image and changing its values with random numbers. Finally, 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 pairs drawn randomly from training data, while $\lambda \sim Beta(\alpha, \alpha)$. Therefore, parameter α affects regularization strength (larger α implies stronger regularization). Mixup is encouraging linear behavior between training examples which has other positive side effects. Original paper shows that it also improves robustness to corrupted labels. As discussed previously, majority of 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 handle label noise with minimal additional computational requirements. These desirable properties made mixup crucial part of every pipeline. Regarding parameter value, α between 0.2 and 0.3 was found to be optimal across different architectures. Every augmentation technique is performed during training phase.

3.3. 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 sufficiently diverse set of base predictions, real gains come with stacking. Stacking is performed by using predictions as features for meta-model. It is often done in cross-validation setting, but because of train-validation split used by level-1 models, we are constrained to use only 10% of data for meta-model training. Predictions of individual classifiers are stacked in columns for both validation and test set. For example, each of ten models would have 41 (number of classes) predictions per example and resulting feature matrix would have 410 columns. Validation data then becomes new training set and stratified 5-fold cross-validation is used for training. Experiments have shown that logistic regression is suitable candidate for meta-classifier. Principal component analysis (PCA) is used to reduce dimensionality of data in each of K iterations. Predictions of meta-model on validation data are combined and MAP score is computed to produce new performance estimate. Finally, trained meta-model is used to generate test set predictions.

4. RESULTS

In this section, results of proposed architectures are presented and discussed. Table 3 summarizes important information regarding models and their scores on validation dataset:

Table 3: Models, preprocessing and scores

Id	Length	Overlap	Transformation	MAP
1	4s	1s	Mel-spec 96 bands	0.916
1	5s	1s	CQT 110 bins	0.942
1	6s	1s	Mel-spec 96 bands	0.919
1	8s	2s	CQT 110 bins	0.929
2	4s	1s	CQT 96 bins	0.945
2	5s	1s	Mel-spec 96 bands	0.926
3	4s	1s	CQT 96 bins	0.901
4	4s	1s	CQT 96 bins	0.946
5	4s	1s	CQT 96 bins	0.93
5	5s	1s	CQT 96 bins	0.927
6	4s	1s	CQT 110 bins	0.955

For clarity, models are specified only through ids, with respect to order of presentation. We can come up with several interesting conclusions by inspecting these values. Experiments have shown that CQT outperforms mel-spectrogram for most models. Also, inputs of length 4s and 5s seem to be optimal, while larger number of bins certainly helps, but also increases computation time. Because of limited number of submissions, some base models were not evaluated on public leaderboard, but it's worth noting that leaderboard score was consistently lower for submitted predictions. Since validation setup has proven to be directionally stable (lower score on validation, usually meant lower score on public leaderboard and vice versa), predictions were mostly submitted after ensembling.

Organizers provided baseline model for comparison with proposed solutions. Its inputs are log scale mel-spectrograms with windows of length 0.25s and hop size of 0.125s. Model contains

3 convolutional layers before output softmax layer. It achieves MAP of approximately 0.704 on public leaderboard.

Final solution is ensemble of 11 proposed configurations. Meta model is logistic regression with regularization parameter $C = 4$. It is trained on 120 features, after PCA dimensionality reduction. It achieves MAP of 0.971 on validation data and 0.956 on public leaderboard, which is an improvement over level-1 models and baseline.

5. CONCLUSION

This paper proposes ensemble of convolutional neural networks for classification of general audio signals. We introduced several architectures, preprocessing techniques and validation setup in order to get diverse set of base predictions. Logistic regression is then used as meta-model to obtain 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.

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"

[20] https://keras.io/

[21] Z. Zhong, L. Zheng, G. Kang, S. Li, Y. Yang, "Random erasing data augmentation

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