DIVERSIFIED SYSTEM OF DEEP CONVOLUTIONAL NEURAL NETWORKS WITH STACKED SPECTRAL FUSION FOR AUDIO TAGGING

Ria Chakraborty[*]

IBM India Pvt. Ltd.
Plot XI-7, Block EP/GP, Sector V, Salt Lake
Kolkata, 700094, INDIA
rchakra1@in.ibm.com

ABSTRACT

This paper outlines a diversified system of deep convolutional neural networks with stacked fusion of spectral features for the DCASE 2018 Task 2 [1], freesound general-purpose audio tagging.

The primary objective of this research has been to design a solution which can churn out decent performance and be deployed within reasonable resource constraints. The two best performing submissions are the results of only two and three different CNNs, with their results being combined based on a boosted tree algorithm with fused spectral features. Experimental results show that the proposed system and preprocessing methods effectively learn acoustic characteristics from the audio recordings, and their ensemble models significantly reduce the error rate further, exhibiting a MAP@3 score of 0.933 and 0.932. This solution has been ranked 21st in the Leaderboard.

Index Terms— Deep Learning, DCASE 2018 challenge, Audio Tagging, frequency-delta augmentation, boosted tree

1. INTRODUCTION

The application areas of Artificial Intelligence (AI) in the domain of audio are endless. For example, consider a smart device which can monitor its surroundings for raising necessary alarms during emergencies; say domestic violence, breaking and entering, etc. Or in the case of a self-driving car where sound-event detection can augment the safety protocols. AI can also contribute towards an enhanced experience involving voice communications with someone experiencing a hearing impairment by using techniques for transcribing speech to text in real-time. For all the above to work, such systems need to combine multiple functionalities involving the audio signals.

In this challenge, a small part of the big picture was presented – tagging the sounds to their appropriate classes. This paper is organized into several sections which outline the details of the dataset, the feature engineering process and in the end, the models consuming the extracted features to correctly identify the class of each input sound.

[*] Primary and single author.

2. THE DATASET

The DCASE 2018 task 2 challenge dataset [2] was provided by Freesound. This Dataset is an audio dataset containing 18,873 audio files annotated with labels from Google's AudioSet Ontology.

The provided sound files are uncompressed PCM 16-bit, 44.1 kHz, mono audio files with widely varying recording quality and techniques. The provided sounds are unequally distributed in 41 categories of the AudioSet Ontology.

3. DATA PRE-PROCESSING AND AUGMENTATION

As a first step, silence removal was performed from the audio files. The removal was performed from both the ends and in-between. The threshold was set at 60 dB. Each file was split into non-silent chunks and then recombined.

As a second step, the sound files were down sampled at a sampling rate of 22050 Hz and split into 4 seconds chunks with a 10% overlap. A second set was prepared without the down sampling operation. Even though arguments exist against the chunking process and inheritance of labels by each chunk, this step became necessary since the audio files were quite long, due to which attention-based hybrid models leveraging both spatial and temporal patterns, alternatively popularized as convolutional recurrent neural networks [3], [4] were extremely slow and required huge computational resources.

However, keeping in mind that Audio events may occur for a brief period in a recording, and the chunks with inherited labels may indeed introduce some incorrect tags, a data augmentation method called Mixup [5] was employed. The next section describes this data augmentation method, which attends to the fact that labels are noisy. This process helped learning of robust features which reduced the overfitting.

3.1. Mixup

Large deep neural networks are powerful but exhibit undesirable behaviors such as memorization and sensitivity to adversarial examples. Mixup [5] is a simple learning principle to alleviate these issues. Essentially, Mixup trains a neural network on con-

vex combinations of pairs of examples and their labels. By doing so, it regularizes the neural network to favor simple linear behavior in-between training examples.

In a nutshell, Mixup constructs virtual training examples.

$\tilde{x} = \lambda x_i + (1 - \lambda)x_j$, where x_i , x_j are raw input vector (1)

$\tilde{y} = \lambda y_i + (1 - \lambda)y_j$, where y_i, y_j are one-hot label encodings (2)

(x_i,y_i) and (x_j,y_j) are two examples drawn at random from our training data, and $\lambda \in [0,1]$. The λ parameter is drawn from Beta (α, α), for $\alpha \in (0, \infty)$. For the submitted results, α was set to 1.

Therefore, Mixup extends the training distribution by incorporating the prior knowledge that linear interpolations of feature vectors should lead to linear interpolations of the associated targets. Mixup can be implemented in a few lines of code and introduces minimal computation overhead. Despite its simplicity, this allows new state-of-the-art performances by increasing the robustness of neural networks when learning from corrupt labels. However, it was observed that Mixup makes the minimization of training loss difficult.

4. FEATURE ENGINEERING

The first step in any sound recognition system is to extract features, i.e. the components of the audio signal that are good for identifying the content.

Following the successful research activities regarding raw-waveform based models [6], [7], experimentation with a 1-D CNN with 2 seconds of raw waveform data as input was performed. However, the network took too long to converge. Hence this approach was not included in the final submission, although, this did learn some useful features. The next focus was on analyzing the frequencies in the data.

To extract the frequency contours in the signal, a shift from the amplitude over time to the frequency over time representation is required. Such a representation is commonly known as spectrogram. Under this process, the signal gets sliced into overlapping frames and a window function is applied to each frame; afterwards, a Fourier transform on each frame (or more specifically a Short-Time Fourier Transform) is performed and the power spectrum is calculated. Subsequently mel filter banks are applied to the power spectra, energies are summed and expressed in the log scale.

The presented solutions use three different sets of features as described below:

- Feature Set – I: For each 4 second chunked clip with 10% overlaps, this set features a sliding window with a length of 2 seconds with 50% overlap. Sampling rate was set at 22050 Hz. The slightly longer window choice was inspired by the performance reported in [8]. The number of mel frequency bands was set at 150, with highest frequency clipped at 8000 Hz. Afterwards, the deltas and delta-deltas were also computed with a width of 9 and supplied as separate channels in the input to the network. These provided a local estimate of the derivative of the input data along the columns. Hence, the dimensions of this set of features are 150 x 150 x 3.
- Feature Set – II: For the other set of chunks without the down sampling operation, spectrograms were extracted with a frame length of 80ms and hop length of 10ms. Number of mel-frequency bands were set at 64. The dimensions for this set is 64 x 401. Even though the final Feature Set – II didn't employ down sampling, experimentation results with 22050 Hz didn't suggest performance degradation.
- Feature Set – III: This set of features were extracted directly from the waveforms. Apart from computing the basic summary statistics of the audio files like min, max, skewness, kurtosis etc., this set also includes summaries of the spectral features like centroid, bandwidth, rolloff, flatness, zero-crossing-rate and 12 Mel Frequency Cepstral Coefficents,

Together, these three sets of features enabled the system to look at the signal from different perspectives, e.g. over both longer and shorter time slices. Further, experimental results revealed that Feature Set – I and Model – I (described below) are extremely robust and yield satisfactory performance even without the chunking operation. (Chunking contributed to only ~0.002 positive gain). Most of the above-mentioned feature engineering computations leverage librosa [9] implementations.

5. CLASS IMBALANCE PROBLEM

The provided dataset suffers from class imbalance problem. For example, classes like 'Glockenspiel' and 'Scissors' have only 94 and 95 samples available, whereas classes like 'Tearing' and 'Violin_or_fiddle' have 300 samples each. That is nearly 3 times the rare classes. To counter this problem, for Level 1 models, the 'class_weight' parameter in Keras [10] was supplied with the class proportion information. This is to inform the model to "pay more attention" to samples from an under-represented class. For the level 2 model, XGBoost, the 'weight' parameter in the DMatrix() function was leveraged.

6. VALIDATION SCHEME

Since the evaluation metric for the leaderboard is MAP@3 and the number of observations had been inflated, a separate validation scheme was designed to estimate the model performance. The training data was split into different number of folds (5, 5 and 4 for the three models respectively), and their out-of-sample predictions were used to estimate the MAP@3 and accuracy scores for both the verified and verified-non-verified mix. Better model performances under this scheme translated mostly in a correlated manner to the public LB performance as well. These out-of-fold predictions were later stacked and used to train a level 2 model, along with the spectral features (Feature Set – III) for generating the ensembled predictions, as described in later sections.

7. LEVEL 1 AND LEVEL 2 MODELS

The best performing solutions leverage a total of two and three different deep convolutional neural networks (DCNN) as the

level 1 models, the details of which are given below. The predictions from these level 1 models are further fused with the spectral features (Feature Set – III) and fed into a level 2 model.

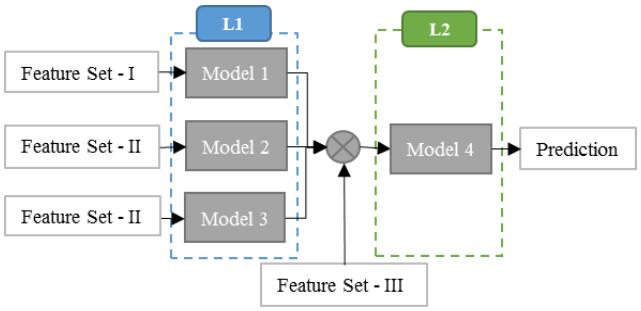

Figure 1: Deep Convolutional Neural Networks serving as level 1 models, with stacked fusion of spectral features for the level 2 classifier.

7.1. Model – I: VGG Style (Level 1)

Input to this network are the mel-spectrograms from Feature Set-I, which are of shape 150 x 150 x 3. Each channel houses the original, delta and delta-delta features of width 9 respectively. The architecture is as follows:

Conv2D, stride=2, kernel_size=5, filters=32, Batch Normalization, ReLU Conv2D, stride=1, kernel_size=3, filters=32, Batch Normalization, ReLU Max Pooling, pool_size=2, Dropout (0.3)
Conv2D, stride=1, kernel_size=3, filters=64, Batch Normalization, ReLU Conv2D, stride=1, kernel_size=3, filters=64, Batch Normalization, ReLU Max Pooling, pool_size=2, Dropout (0.3)
Conv2D, stride=1, kernel_size=3, filters=128, Batch Normalization, ReLU Conv2D, stride=1, kernel_size=3, filters=128, Batch Normalization, ReLU Conv2D, stride=1, kernel_size=3, filters=128, Batch Normalization, ReLU Conv2D, stride=1, kernel_size=3, filters=128, Batch Normalization, ReLU Max Pooling, pool_size=2, Dropout (0.3)
Conv2D, stride=1, kernel_size=3, filters=512, Batch Normalization, ReLU Dropout (0.5)
Conv2D, stride=1, kernel_size=1, filters=512, Batch Normalization, ReLU Dropout (0.5)
Conv2D, stride=1, kernel_size=1, filters=41, Batch Normalization, ReLU Global Average Pooling, Softmax

Table 1: Network architecture for the DCNN Model – I: VGG Style

The feature learning section in the network is inspired from VGGNet [11], and the classification section leverages network-in-network structure with 1x1 convolutions over the volume. This made the network considerably deep and narrow. With 11 convolutional layers and added batch normalizations, this network features less than 1.5M parameters. The optimizer was minibatch gradient descent with a variant of Adam based on the infinity norm.

The data was split into 5 folds, and their performances for the test set were aggregated using geometric averaging. For scaling, mean and standard deviation statistics were calculated based on the training set for each iteration (using the 4 folds), and these values were used to transform the validation (1 fold) and test set. The data was scaled for each channel separately.

7.2. Model – II: Half-portion AlexNet featuring Cyclic Learning Rate (Level 1)

Slightly bigger than the first one (~1.7M parameters), this model uses the Feature Set – II as its input. The classic AlexNet [12] performs delineation of responsibilities between the two GPUs by separating computations in two portions. This model is based on half-portion of AlexNet. In addition to Mixup augmentation, this net used some more augmentations like randomly shifting images (spectrograms) horizontally and random cut-outs/erasing [13].

The optimization process leveraged cyclic learning rate [14] policy. This method for setting the learning rate eliminates the need to experimentally find the best values and schedule for the global learning rates. Instead of monotonically decreasing the learning rate, this method lets the learning rate cyclically vary between reasonable boundary values. Training with cyclical learning rates instead of fixed values achieves improved classification accuracy without a need to tune and often in fewer iterations. The mode was set to 'triangular' for this.

This model was trained on 5 folds and the results were geometrically averaged.

7.3. Model – III: SE-ResNet-50 with Cyclic Learning Rate (Level 1)

This is the heaviest model in this solution, featuring a whooping ~26M number of parameters! This is an implementation of Residual Networks (ResNets) featuring "Squeeze-and-Excitation" blocks [15]. "Squeeze-and-Excitation" (SE) block adaptively recalibrates channel-wise feature responses by explicitly modelling interdependencies between channels. SE-ResNet-50 is a specific configuration referenced from the paper.

With Feature Set – II as input, this model used mixup, random cut-out/erasing and horizontal shift data augmentations, along with Cyclic learning rates. The model was trained on 4 folds and the results were geometrically averaged.

7.4. Model – IV: XGBoost (Level 2)

The out-of-fold predictions for the train set were stacked together and along with the spectral features, this became the new feature set for the level 2 classifier. Likewise, the test predictions from the level 1 models were stacked and the new features for

the test set were created. The level 2 model was chosen to be XGBoost [16]. XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. Predictions from the level 1 models were aggregated to original file level before being fed into the L2 model.

This model featured shallow trees, with 'max_depth' parameter set to 3. Learning rate was set at 0.03; 'subsample' and 'colsample_bytree' values were set at 0.7. 4 folds were used to assess the performance on the train set and the results were finally aggregated using geometric averaging.

8. RESULTS

This section describes the performance of the best performing solutions, along with the individual modules.

8.1. Submission description

The two different submissions are based on the methods described in the previous sections for the DCASE-2018 challenge, Task 2. They are:

1. Chakraborty_IBM_Task2_1: Stacked and fused ensemble of Model – I: VGG Style, Model – II: Half-portion AlexNet featuring Cyclic Learning Rate and Model – III: SE-ResNet-50, with XGBoost as the L2 classifier.
2. Chakraborty_IBM_Task2_2: Simple average of the Model – I and Model – III results.

The Model – I in conjunction with the feature representation in Feature Set – I was found to be extremely effective. The comparisons are highlighted below:

Network Description	Public LB	Private LB	Overall
Model - I	0.925	0.908	0.911
Model - II	0.926	0.904	0.908
Model – III	0.939	0.909	0.915
Chakraborty_IBM_Task2_1	0.947	0.933	0.936
Chakraborty_IBM_Task2_2	0.945	0.932	0.934

Table 2: Model performance comparison on verified labels only

Class-level score distribution for the best performing submission, split into two tables:

Class	MAP@3	Class	MAP@3
Oboe	0.984	Drawer_open_or_close	0.879
Bass_drum	1.000	Snare_drum	0.956
Saxophone	0.973	Fart	1.000
Chime	0.879	Meow	0.983
Electric_piano	0.932	Trumpet	0.955
Shatter	0.966	Fireworks	0.714
Bark	0.964	Bus	0.833
Acoustic_guitar	0.907	Keys_jangling	0.833
Scissors	0.693	Applause	1.000
Double_bass	0.988	Harmonica	0.924

Table 3: MAP@3 scores for classes - I

Class	MAP@3	Class	MAP@3
Knock	0.970	Cough	1.000
Telephone	0.781	Gong	0.946
Violin_or_fiddle	0.995	Glockenspiel	0.828
Gunshot_or_gunfire	0.905	Tearing	0.982
Burping_or_eructation	1.000	Writing	0.908
Clarinet	1.000	Squeak	0.644
Computer_keyboard	0.897	Microwave_oven	0.925
Flute	0.961	Laughter	0.987
Cello	0.960	Finger_snapping	1.000
Tambourine	0.983	Hi-hat	0.944
		Cowbell	0.988

Table 4: MAP@3 scores for classes - II

9. CONCLUSION

In this paper, different solutions are presented with different configurations, along with their parameters. The submissions use three different CNNs for generating the level 1 predictions. These predictions are stacked and fused with the spectral features to form a new feature set, which is trained using a level 2 model, chosen to be XGBoost. This prediction scores 0.933 in the private LB. A simple average of the Model – I and Model - III scores 0.932 in the private LB.

The Model – I, along with Feature Set – I can generate good predictions. Its ensemble with any other solution makes more robust predictions, at the expense of an increase in the number of parameters. Together with strong data augmentation techniques, like Mixup and random erasing/cut-outs, more complex models can be trained for an improved performance. Further, improvements around the loss function to accommodate varying levels of penalty for the verified and non-verified labels could push for better scores.

10. REFERENCES

[1] http://dcase.community/workshop2018/.
[2] Eduardo Fonseca, Manoj Plakal, Frederic Font, Daniel P. W. Ellis, Xavier Favory, Jordi Pons, Xavier Serra, "General-purpose Tagging of Freesound Audio with AudioSet Labels: Task Description, Dataset, and Baseline", https://arxiv.org/abs/1807.09902
[3] Sharath Adavanne and Tuomas Virtanen. Sound event detection using weakly labeled dataset with stacked convolu-

tional and recurrent neural network. In DCASE Workshop, 2017.

[4] Yong Xu, Qiuqiang Kong, Wenwu Wang, Mark D. Plumbley, "Large-scale weakly supervised audio classification using gated convolutional neural network", https://arxiv.org/abs/1710.00343

[5] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, David Lopez-Paz., "mixup: Beyond Empirical Risk Minimization", https://arxiv.org/abs/1710.09412

[6] Jongpil Lee, Taejun Kim, Jiyoung Park, Juhan Nam, "Raw Waveform-based Audio Classification Using Sample-level CNN Architectures", https://arxiv.org/pdf/1712.00866.pdf

[7] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In IEEE ICASSP, pages 6964–6968, 2014.

[8] Karol J. Piczak, "Environmental Sound Classification with Convolutional Neural Networks", http://karol.piczak.com/papers/Piczak2015-ESC-ConvNet.pdf

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

[10] https://keras.io/

[11] Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", https://arxiv.org/abs/1409.1556

[12] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, "ImageNet classification with deep convolutional neural networks", NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 Pages 1097-1105

[13] Yusuke Uchida, "Cutout / Random Erasing implementation, especially for ImageDataGenerator in Keras", https://github.com/yu4u/cutout-random-erasing

[14] Leslie N. Smith, "Cyclical Learning Rates for Training Neural Networks", https://arxiv.org/pdf/1506.01186.pdf

[15] Jie Hu, Li Shen, Gang Sun, "Squeeze-and-Excitation Networks", https://arxiv.org/pdf/1709.01507.pdf

[16] https://xgboost.readthedocs.io/en/latest/