Music Auto-Tagging

1. Introduction

One of the main advantages of the advancement of Music Information Retrieval (MIR) is the fact that it has become possible to achieve tailored music enjoyment. Increasing attention to Music Auto-Tagging (MAT) research, in particular, has led to a massive leap in music recommendation and search. MAT refers to a task that assigns tag words to a given song according to its features, such as genre, instruments, vocals and beats, and this is beneficial for listeners to grasp target songs by linguistically visualized characteristics.

Numerous studies have proposed an assortment of neural network architectures. Choi et al. [1] suggested a framework that fully consists of convolutional neural networks, varying the number of layers, pooling strategies, and input data format. The authors adopted two different datasets, *MagnaTagATune* (MTAT) [2] and *Million Song Dataset* (MSD) [3], showing that the experiment of 4 layers of fully convolutional networks (FCN) taking Mel-spectrogram as input represents the highest prediction performance on MAT, and a case that comprises of 6 layers of FCN outperforms on MSD. Choi et al. [4] proposed a method taking advantage of recurrent neural networks (RNN) as a summarizer for temporal representation at the end of convolution layers, which is called CRNN. They experimented performances from several model frameworks, indicating CRNN resulted in better performance than others (ex. k2c1, k1c2, and k2c2) given with the same number of parameters, but it also shows the lowest training speed, which is considered as one of the major shortcomings in an operation of RNN. Additionally, an approach that makes use of raw audio signal as input has been adopted in [5, 6, 7], and it is known to be advantageous to mitigate the effort of data pre-processing and learn hierarchical attributes effectively. Specifically, the authors [5] suggested a DCNN (Deep CNN, at least 7 layers) framework and experimented it with not just frame-level raw audio data but a very small amount of samples, and compared them to state-of-the-art performance from previous works. Results indicate the proposed method appeared to perform comparable prediction accuracy as generic network frameworks that took Mel-spectrogram as input, and even outperform with a large dataset, MSD.

In this paper, I explore a couple of model frameworks mentioned before and analyze each performance result in a point of view of model's architecture and hyperparameter adjustment. At the end of this work, I hope I could have a better understanding of how neural networks work.

2. Data and Methods

2.1 Dataset

Several datasets have been introduced [2, 3, 8], and the commonly used one in MAT tasks is MTAT which is characterized as a weakly labeled dataset and consists of 25,863 clips of 29 seconds long mp3 file (16kHz of sampling rate) from 446 albums of 230 artists. It also contains 188 tag words describing genres, instruments, mood, and vocals, but upmost 50 frequently appeared tags are used in this work.

2.2 Experiment I: Tune hyperparameters

The first experiment targets on determining the optimum hyperparameter combination. Results that neural networks spawn are highly susceptible to modifying hyperparameters, and it should be determined depending upon research goals. For instance, widening the window size for STFT (Shore-Time Fourier Transform) is beneficial to detect a more accurate pitch level, but it also has a negative influence on defining the exact time a certain note is on an onset state. Moreover, deep hidden layers are helpful for neural networks to break down the input feature map in a variety of perspectives, but it is likely to cause an increase in model's complexity, with performance saturating to a certain point. Therefore, experimentations for optimizing hyperparameters are preceded, and this procedure particularly focuses on tuning FFT size, number of mel bins, depth of hidden layers, and number of convolution layers [Table 1].

	Model	# of Conv. layers	FFT size	Hidden size	# of Mel bins	Accuracy	PR-AUC	ROC-AUC
Ex. 1	Control	3	1024	64	48	0.9393	0.3353	0.8568
	Type A	3	2048	64	48	0.9391	0.3350	0.8556
	Type B	3	1024	128	48	**0.9403**	**0.3986**	**0.8791**
	Type C	3	1024	64	96	0.9391	0.3388	0.8588
	Type D	4	1024	64	48	0.9392	0.3097	0.8374
	Type E	3	1024	128	96	**0.9417**	**0.4166**	**0.8846**
	Type F	4	1024	128	96	0.9408	0.3998	0.8821
	Type G	5	1024	128	96	0.9404	0.3958	0.8839

Table 1. Validation results of different hyperparameter schemes

2.3 Experiment II: Model architecture

The main purpose of this experiment is to identify performance differences among distinct frameworks, specifically concentrating on convolution layer dimensionality and the final layer. Conventional MAT task is accompanied by a single fully connected layer at the end of convolution layers to extract global features [9]. However, it is a renowned fact that it is inclined to result in overfitting because of all neurons being allocated to each input data [1]. Due to the major shortcoming of multi-layer perceptron, researchers have suggested various model frameworks, and this work takes FCN [1] and CRNN [4] into account. It is worth to note that every convolution layer in the first experiment is comprised of 1D kernels, whereas this experiment deals with 2D convolution because of the presence of potential harmonic features and direct comparison to previous research [Table 2].

2.4 Performance examination

Since MAT is categorized into a multi-hot encoding classification, Binary Cross Entropy (BCE) is an appropriate loss function. The formula is following:

$$BCE = -\frac{1}{n}\sum_{i=1}^{n}(y_i \cdot log\,\hat{y}_i + (1 - y_i) \cdot log(1 - \hat{y}_i))$$

where n, y and ŷ represents given batch size, target labels and prediction, respectively.

Assorted statistical methods to measure model's performance are used:

$$Accuracy = \frac{TP + TN}{P + N}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{P}$$

$$FPR = \frac{FP}{FP + TN}$$

Each provides a distinctive implication, where Accuracy shows the number of any correct predictions considering the whole ground truth. However, Precision does not take into account model's negative predictions but focuses on how many positive predictions are correct, which indicates that it is useful to identify performances in a specific class. On contrary, Recall, also called True Positive Rate (TPR), does not consider any negative ground truth, representing the fact that it concentrates on correct positive predictions out of all positive ground-truth. Both Precision and Recall have in common in terms of targeting a certain class, but they can be differentiated by which standpoint we would like to be on and look at correct positive predictions. A usual presentation is to compute the area under the Precision-Recall curve (PR-AUC), and its value ranges within [0, 0.5], where 0.5 denotes a

Ex. 2	1D FCN			2D FCN			2D CRNN		
	C = 3	C = 4	C = 5	C = 3	C = 4	C = 5	C = 3	C = 4	C = 5
Accuracy	0.9443	0.9448	0.9438	**0.9492**	0.9457	0.9460	0.9406	0.9444	0.8520
PR-AUC	0.4779	0.4906	0.4870	0.4433	0.5048	**0.5161**	0.3773	0.4851	0.3952
ROC-AUC	0.9018	0.9069	0.9063	0.8963	0.9160	**0.9191**	0.8749	0.9093	0.8897

Table 2. Validation result of different model architectures. C indicates the number of convolution layers.

perfect guess. Another similar metric that commonly tags along with PR-AUC is ROC (Receiver Operator Characteristics)-AUC, which is calculated by the ratio between TPR and False Positive Rate (FPR). The main difference between them lies in the robustness of reflecting skewed datasets [12]. Since the majority of labels in the MTAT are zeros, FPR is unable to provide a performance level sorely based on model's predictability. Lastly, F1-score, the harmonic mean of Precision and Recall, is also used.

2.5. General configuration

Consistent environments for equal comparison are achieved by applying log-amplitude Mel-spectrogram as input, ADAM stochastic optimization [10], Batch Normalization before Rectified Linear Unit (ReLU) activation function, Dropout layers with p = 0.5. It is noteworthy that subsampling layers are also added to every convolution layer but with different pooling strategies while adopting the same pooling size throughout the whole attempts in the first experiment. Finally, summarized 50 output vectors are converted into probability representation through Sigmoid function.

3. Results and Discussion

3.1. Analysis on Experiment I

[Table 1] displays performance evaluations from multiple experimentations. First and foremost, the best performance is observed Type B model, 2 times larger hidden size, which indicates that those input data require an enough number of kernels at each layer to be broken down. For neural networks, possessing a large hidden size represents the ability that is able to assign a wide variety of weights to a single input data. Secondly, trivial performance degradation is detected on the Type A model, widening window size for STFT, but it does not cause as much change as modifying the hidden size, which has a connotation that taking a large number of samples for STFT does not have a difficulty to identify features and classify them. Third of all, enlarging the number of mel bins on the Type C model results in a marginally better performance than the control model, demonstrating that either potential features in an expanded mel range contribute to prediction or it is a consequence of different weight initialization. At last, adding one more convolution layer on the Type D model experiences a significant performance decline. To further examine the effect of appending more convolution layers, an additional sub-experiment was conducted with optimal hyperparameters according to previous results, giving rise to the fact that a gradual degradation is found as the increase of the number of convolution layers, and this can be illustrated by the consequence of overfitting on the training dataset.

3.2. Analysis on Experiment II

Experimentations on varying model's architecture [Table 2] result in 5 convolution layers of the 2D FCN model outperforming over other frameworks confirmed by proposed evaluation metrics except for accuracy. The discrepancy of prediction estimations, specifically speaking of accuracy and ROC-AUC, on the model is likely to be caused by FPR, where the majority of positive predictions as the threshold becomes lower must have been considered as false positives owing to the unbalanced dataset. Furthermore, a significant improvement of PR-AUC on the model as deepening convolution layers is observed but not on ROC-AUC as much of it, which spawns speculation that an apparent upgradation is evident in the standpoint of model performance, but it is not satisfied when taking the number of corrected ground-truths into account. Meanwhile,

Test	1D FCN (C=4)	2D FCN (C=5)	2D CRNN (C=4)
Accuracy	0.9423	**0.9425**	0.9419
PR-AUC	0.4949	**0.5207**	0.4863
ROC-AUC	0.9075	**0.9182**	0.9102

Table 3. Test result of different model architecture with optimal setting

4 layers of 1D FCN and 2D CRNN show comparable performances to one another**,** with representing considerable differences on PR-AUC to those of 5 layers of 2D FCN.

To further examine the performances of each best architecture, experiment results with the test dataset are shown on the [Table 3]. Negligible accuracy differences that are likely to be derived from variable weight initialization appear, and the existence of a substantial gap in PR-AUC and a marginal ROC-AUC against each model still remains. These results give rise to the conclusion that 2-dimensional convolution layer architecture is the optimal strategy to automatically predict music tags, which is in agreement with previous research [1].

4. Conclusion

In this paper, I provided results of the prediction performance of several different models with adjusting hyperparameters and architectures. I came to the end with a verification that comprising neural networks only 2D fully convolution layers is the ideal option for MAT based on a couple of experiments presented before. Additionally, accuracy is able to denote the rate of correctly predicted ground truths out of the whole labels, but it may not be a perfect metric when dealing with a skewed dataset. Moreover, ROC-AUC is an excellent metric for statistical measurement, nevertheless, PR-AUC is also essential to analyze performances in a model's viewpoint.

5. References

[1] Keunwoo Choi, György Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[2] Edith Law, Kris West, Michael I. Mandel, Mert Bay, and J. Stephen Downie. Evaluation of algorithms using games: the case of music tagging. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, 2009.

[3] Thierry Bertin-Mahieux, Daniel PW. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, 2011.

[4] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *Proceedings of 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[5] Jongpil Lee, Jiyoung Park, Keunhyoung Luke Kim, and Juhan Nam. Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms. In *Sound and Music Computing Conference (SMC)*, 2017.

[6] Jordi Pons, Oriol Neito, Matthew Prockup, Erik Schmidt, Andreas Ehmann, and Xavier Serra. End-to-end learning for music audio tagging at scale. In *proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[7] Zhenyao Zhu, Jesse H. Engel, and Awni Hannun. Learning multiscale feature directly from waveforms. *arXiv preprint arXiv:1603.09509*, 2016.

[8] D. Bogdanov, M. Won. P. Tovstogan, A. Porter, and X. Serra. The mtg-jamendo dataset for automatic music tagging. In *Machine Learning for Music Discovery Workshop, International Conference on Machine Learning (ICML)*, 2019.

[9] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In

Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[12] Jesse Davis and Mark Goadrich. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 2006.

1D FCN (C=3)	1D FCN (C=4)	1D FCN (C=5)
Mel-spectrogram (Input: 96 × 911)		
Conv. 1×3×128 (# of params: 3.7×10^4)	Conv. 1×3×128 (# of params: 3.7×10^4)	Conv. 1×3×128 (# of params: 3.7×10^4)
MP (3) (Output: 128 × 303)	MP (3) (Output: 128 × 303)	MP (2) (Output: 128 × 454)
Conv. 1×3× 256 (# of params: : 9.9×10^4)	Conv. 1×3× 256 (# of params: 9.9×10^4)	Conv. 1×3× 256 (# of params: 9.9×10^4)
MP (3) (Output: 256 × 100)	MP (3) (Output: 256 × 100)	MP (2) (Output: 256 × 226)
Conv. 1×3× 512 (# of params: 4.0×10^5)	Conv. 1×3× 512 (# of params: 4.0×10^5)	Conv. 1×3× 512 (# of params: 4.0×10^5)
MP (3) (Output: 512 × 32)	MP (3) (Output: 512 × 32)	MP (3) (Output: 512 × 74)
	Conv. 1×3× 1024 (# of params: 1.5×10^6)	Conv. 1×3× 1024 (# of params: 1.5×10^6)
	MP (3) (Output: 1024 × 10)	MP (3) (Output: 1024 × 24)
		Conv. 1×3× 2048 (# of params: 6.3×10^6)
		MP (3) (Output: 2048 × 7)

Supplementary 1. Architecture of 1D FCN

2D FCN (C=3)	2D FCN (C=4)	2D FCN (C=5)
Mel-spectrogram (Input: 1 × 96 × 911)		
Conv. 3×3×128 (# of params: 1.5×10^3)	Conv. 3×3×128 (# of params: 1.5×10^3)	Conv. 3×3×128 (# of params: 1.5×10^3)
MP (3, 3) (Output: 128× 31 × 303)	MP (2, 3) (Output: 128× 47 × 303)	MP (2, 2) (Output: 128× 47 × 454)
Conv. 3×3× 256 (# of params: 2.9×10^5)	Conv. 3×3× 256 (# of params: 2.9×10^5)	Conv. 3×3× 256 (# of params: 2.9×10^5)
MP (3, 3) (Output: 256 × 9 ×100)	MP (3, 3) (Output: 256 × 15 ×100)	MP (2, 2) (Output: 256 × 22 × 226)
Conv. 3×3× 512 (# of params: 1.2×10^7)	Conv. 3×3× 512 (# of params: 1.2×10^7)	Conv. 3×3× 512 (# of params: 1.2×10^7)
MP (3, 3) (Output: 512 × 2 × 32)	MP (3, 3) (Output: 512 × 4 × 32)	MP (2, 3) (Output: 512 × 10 × 74)
	Conv. 3×3×1024 (# of params: 4.7×10^7)	Conv. 3×3×1024 (# of params: 4.7×10^7)
	MP (4, 3) (Output: 1024× 1 × 10)	MP (2, 3) (Output: 1024× 4 × 24)
		Conv. 3×3× 2048 (# of params: 1.8×10^8)
		MP (2, 3) (Output: 2048 × 1 × 8)

Supplementary 2. Architecture of 2D FCN

CRNN (C=3)	CRNN (C=4)	CRNN (C=5)
Mel-spectrogram (Input: $1 \times 96 \times 911$)		
Conv. 3×3×128 (# of params:)	Conv. 3×3×128 (# of params:)	Conv. 3×3×128 (# of params: 1.5×10^3)
MP (3, 3) (Output: 128×23×101)	MP (2, 2) (Output: 128× 47 × 454)	MP (2, 2) (Output: 128× 47 × 454)
Conv. 3×3× 256 (# of params:)	Conv. 3×3× 256 (# of params:)	Conv. 3×3× 256 (# of params: 2.9×10^5)
MP (4, 3) (Output: $256 \times 7 \times 11$)	MP (3, 2) (Output: $256 \times 15 \times 226$)	MP (2, 2) (Output: $256 \times 22 \times 226$)
Conv. 3×3× 512 (# of params:)	Conv. 3×3× 512 (# of params:)	Conv. 3×3× 512 (# of params: 1.2×10^6)
MP (5, 3) (Output: $512 \times 1 \times 32$)	MP (3, 3) (Output: $512 \times 5 \times 74$)	MP (2, 2) (Output: $512 \times 10 \times 112$)
RNN (2 layers, 1 hidden_size) (# of params: 117)	Conv. 3×3× 1024 (# of params:)	Conv. 3×3× 1024 (# of params: 4.7×10^6)
	MP (3, 3) (Output: $1024 \times 1 \times 24$)	MP (2, 3) (Output: $1024 \times 4 \times 37$)
	RNN (2 layers, 1 hidden_size) (# of params: 117)	Conv. 3×3× 2048 (# of params: 1.8×10^7)
		MP (2, 3) (Output: $2048 \times 1 \times 18$)
		RNN (2 layers, 1 hidden_size) (# of params: 117)

Supplementary 3. Architecture of 2D CRNN