END-TO-END LEARNING FOR MUSIC AUDIO

Rohak Roy

University of Bristol
Faculty of Engineering
75 Woodland Rd, Bristol BS8 1UB

***Abstract*—Traditionally, approaches to music information retrieval (MIR) tasks have heavily relied on prior task-specific expertise along with a great deal of engineering efforts to be solved. To reduce the necessary technical effort and the requirement for prior knowledge, there has been a growing interest in adopting feature learning and deep architectures in recent years. In most cases, unfortunately, even this type of method typically depends on mid-level representations of music audio, such as spectrograms, which is a form of pre-requisite expertise. In this work, we try to replicate and further improve Dieleman and Schrauwen's model which explores the possibility of applying feature learning directly to raw unprocessed audio waveforms.**

Keywords—automatic audio-tagging, convolutional neural networks, feature learning, music information retrieval.

I. INTRODUCTION

Due to the rapid advancement of music technology in recent years, music information retrieval (MIR) tasks have been receiving a lot of attention. Audio-tagging is one such task which can have remarkable advantages. For instance, with a vast dataset of music available on the internet, unlike tags, searching for individual artists using their names or descriptive keywords isn't practical and often yields inadequate results. To perform MIR tasks, traditionally researchers extract features from music audio signals which are then delivered as input to regressor or classifier models. This calls for a significant amount of task-specific expertise and substantial engineering efforts.

In various other disciplines such as computer vision or speech-recognition, researchers are now able to develop models of data requiring little to no prior knowledge. This is because of state-of-the-art feature learning techniques which have emerged in recent years and have become increasingly popular. Consequently, the MIR community has also been paying close attention to feature learning. However, most applications of feature learning with regards to MIR still rely on using mid-level representations of the audio-signal such as a log-scaled representation in the frequency domain like Mel-spectrograms which then further undergo amplitude compression operations. Depending on the task, these representations might be required to be transformed into even more compact forms of audio features. Since this entire pipeline of processes requires considerable acoustic expertise, it is thereby considered to be a form of prior knowledge. In contrast, for computer vision tasks a model can learn features by operating directly on the raw pixel representations of images without the need for any type of data preprocessing.

In this paper, we explore the feasibility of employing a similar approach with regards to recreating and further improving a one-dimensional convolutional neural network which is capable of directly receiving raw audio waveforms as input and using them to apply feature learning. This has the potential to drastically reduce the amount of pre-requisite expertise required. Accurately performing automatic audio-tagging tasks is the main aim of this work. The following sections III to X display the base model architecture, discuss its implementation and the dataset it was used to train on, provide results used to assess its performance, discuss the implementation of improvements made to the base model, and finally concludes with future work.

II. RELATED WORK

A previous work [1] focuses on a very similar task wherein they also attempt to increase the accuracy of automatic music audio-tagging using raw waveforms as input to CNN models. They begin by identifying a few issues with Dieleman and Schrauwen's model, which is in fact recreated in this work. These issues are used to justify its shortcomings with respect to not being able to achieve results comparable to models that use Mel-spectrograms as input instead. Some of the issues highlighted include the CNN being limited by an insufficient number of layers and filters, which makes it difficult to learn the complex structure of polyphonic music, its inability to identify suitable non-linear activation functions which can replace the amplitude compression techniques used in spectrograms, and finally the bottom layer taking only frame level inputs which hamper its capacity to learn all the possible phase variations of periodic waveforms which are prevalent in musical signals. To address these issues, this paper provides the raw waveforms in sample-level as input to the network instead. They compare this small granularity to pixel-level in image classification. This paper demonstrates that decreasing the strides of the first convolutional layer from frame-level to sample-level and increasing the depth of the network is proportional to its performance accuracy. The model developed in this paper uses a pair of convolutional layer and max pooling layer of the same size as basic building modules of the CNN. The last convolutional layer in the model is set to a length of 1 reducing the number of parameters between itself and the output layer as well as allowing the model performance to be examined more accurately by the depth of the network and the stride of the first convolutional layer. The network is provided with 59049 samples as input and different combinations of m^n-CNN models are developed where m refers to the filter and pooling length and n refers to the number of layers (depth). Their 3^9 model achieves an AUC score of 0.9055 on the MagnaTagATune dataset which is comparable to previous state-of-the-art performances.

Other similar studies [2] also aim to improve the architecture of sample level one-dimensional CNN models for audio-tagging by providing raw waveforms as input to the network.

This is done by exploring the use of different combinations of building blocks from image classification models: ResNets and SENets. A multi-level feature aggregation layer is added to the end of the network and how this affects performance is observed in the paper. The purpose of this layer is to combine several hidden layer representations for the final prediction. The SE blocks perform a squeeze operation where the temporal dimensionality is reduced to 1 by averaging outputs from each channel. This is followed by an excitation operation where the channel dimension C is first expanded by a factor 'α' via a fully connected layer and then reduced back to its original dimension C through a second fully connected layer before finally being passed through a sigmoid activation function. Res-n blocks are used here to represent blocks which contain n convolutional layers with the addition of a dropout layer to avoid overfitting. They also consist of a skip connection inspired from ResNets. The multi-level feature aggregation reduces the temporal dimensions of the outputs of the last three blocks by a global max pooling. These three outputs are then concatenated and delivered into the last two FC layers. The impact of the multi-level aggregation was proven effective and the combination of these two building blocks referred to as the ReSE-2 block provides an AUC score of 0.9113% on the MagnaTagATune dataset which is comparable to previous state-of-the-art performances using frequency representations.

Another paper [3] looks at achieving accurate audio-tagging but this time using EfficientNet, a family of models that scale depth, width, and resolution consistently. However, in this study the raw audio signal is converted into a Mel-spectrogram which is provided as input to the network. Models EfficientNet B0 to B7 are evaluated where each model comprises of 7 blocks, but the number of sub-blocks within these blocks grow as we progress from B0 to B7 with B0 consisting of 237 layers in total compared to 813 in B7. Out of them all, B0 achieves an accuracy of 0.93 on test data from the MagnaTagATune dataset.

III. DATASET

The model in this work was trained and validated using one of the most popular audio-tagging databases, MagnaTagATune. Annotated with 188 tags, each sample is a 29-second audio clip that has been resampled at a rate of 12KHz from one of the 5223 songs, 445 albums, and 230 performers that are represented. The top 50 most frequent tags were the only ones used. The training and validation data consisted of 16,963 and 4392 samples respectively.

IV. CNN ARCHITECTURE

The CNN architecture recreated for the base model can be visualized in Figure 1. It consists of 7 layers in total. The first layer is a strided convolutional layer with a single filter, where the length and stride of the filter are received as input parameters to the model. The next four layers consist of two pairs, where each pair is a convolutional layer with 32 filters of length 8 and stride 1, and a max-pooling layer of pooling size 4 and pooling stride 1. This is followed by two fully connected dense layers containing 100 and 50 units respectively. All the convolutions and max pooling were conducted only along the axis representing time and were thus one-dimensional operations.

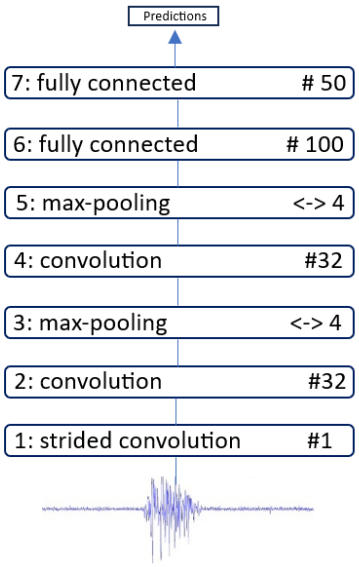

Fig 1. The convolutional neural network architecture for the base model. The raw audio waveform is passed into the model as input which then forward propagates its way through the 7 layers. (#) represents the number of filters in convolutional layers and the number of units in the FC layers. (<->) represents the pooling size of the max pool layers.

V. IMPLEMETATION DETAILS

A. Data Preprocessing

During the initial training phase of the model where the raw audio waveforms were delivered as input, it was observed that the model was not able to achieve sufficient accuracy, especially in the cases where larger values of length and stride were passed as input parameters. This could potentially be occurring due to the data values ranging from -32,768 to 32767 thereby making certain features more dominant than others due to their larger magnitude. To address this issue, the training and validation datasets were pre-processed to normalize them prior to commencing training and evaluation of the model. The normalization process involved passing each value in the dataset through a function:

$$f(x_i) = 2 \times \frac{(x_i - x_{min})}{(x_{max} - x_{min})} - 1 \quad (1)$$

Here, x_i is the i^{th} value in the dataset, x_{min} is the lowest and x_{max} is the greatest value in the dataset. As a result, all the values within the dataset were transformed and limited within the range [-1, 1]. This allowed for a significant increase in the model's accuracy due to facilitating its ability to learn the underlying intricate patterns in the data by removing the influence of different scales.

B. Training

Next, training was done using mini-batch gradient descent, with a mini-batch size of 10. Each value in the dataset is a 29s raw audio waveform sampled at a rate of 12KHz which is broken down into 10 sub-clips, where each sub-clip is 3s

long. Hence, a single value in the dataset has dimension [10, 1, 34950]. For a mini-batch size B, the data loaded and delivered into the model would therefore have dimension [B, 10, 1, 34950]. As a first step, the input data is flattened to convert its dimension into [10B, 1, 34950] instead. This is then forward propagated through the next 5 layers of the network up to the second max-pooling layer. The output data of this layer has dimension [10B, 32, F] where F is its feature space. This temporal dimension is reduced to 1 by a global max-pooling i.e., computing the average across F for each of the 32 filters. The data now having a dimension of [10B, 32, 1] is reshaped to [10B, 32] and forwarded into the remaining two FC dense layers. The output of the final FC layer is of dimension [10B, 50] which is reshaped to [B, 10, 50]. Here, 50 represents each of the audio-tags whereas 10 represents each of the sub-clips. To retrieve model predictions, an average is required to be computed across all 50 tags. To do so, the data is transposed along the 1st and 2nd axes to transform its dimension into [B, 50, 10]. Once again, global max pooling is used to find the average of each tag across all 10-subclips thus changing the dimensions into [B, 50, 1]. This is then transposed back into [B, 1, 50] and reshaped to [B, 50] before finally passing it through a sigmoid activation function which provides the final logits of the model. All other layers besides the final FC dense layer uses ReLu (Rectified Linear Unit) as their activation function to capture non-linearity in the data.

C. Hyperparameters and Evaluation

Although SGD with 0.9 Nesterov momentum was initially chosen, it was later observed that Adam performed much better in facilitating training the model and hence was selected as the optimizer. The learning rate was set to 0.001 and the binary cross entropy function was used to calculate loss. For each length and stride, the model was trained for 20 epochs as validation accuracy began to plateau after this point. Validation accuracy was evaluated at the end of every epoch by computing the area under the ROC curve (AUC) for each tag, and then calculating the average across all 50 tags.

VI. REPLICATING QUANTATIVE RESULTS

Length	*Stride*	*AUC (raw audio)*
1024	1024	0.7428
1024	512	0.7702
512	512	0.7718
512	256	0.7931
256	256	0.7827

Table 1. Results for the audio-tag prediction task on MagnaTagATune using the base CNN architecture, for different combinations of filter lengths and strides in the first strided convolutional layer.

VII. TRAINING CURVES

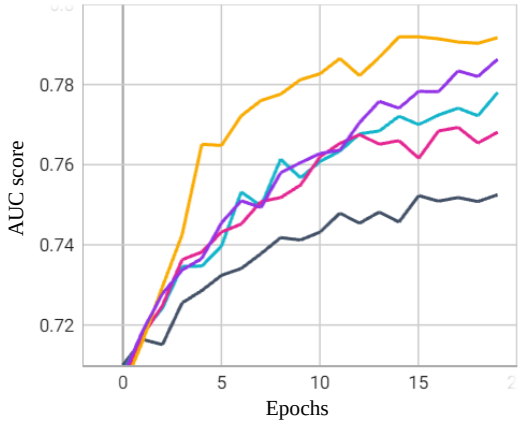

Fig 2. AUC under ROC score evaluated on validation data at the end of every epoch.

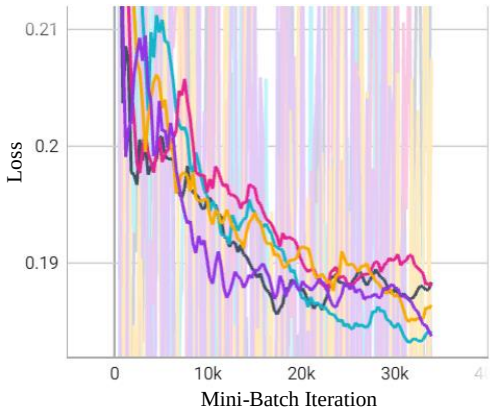

Fig 3. Training loss via binary cross entropy every 300 mini-batch iterations.

Color	Length	Stride
	1024	1024
	1024	512
	512	512
	512	256
	256	256

Fig 4. Legend

In figure 2, the AUC scores for all the combinations of different length and stride input parameters generally show a steady growth. In figure 3, although the mini-batch losses are extremely noisy, we can still conclude all the curves tend to follow a trend of slowly falling as the number of iterations increase.

After around the 10th epoch, most of the AUC curves demonstrate fluctuation with the scores temporarily dropping before quickly rising back up again. However, it is not feasible to imply overfitting to training data from this as training loss also seems to increase momentarily about the same time before falling off again. For these reasons, we do not think any concerning amount of overfitting has occurred during the training phase of the models.

VIII. Qualitative results

Our model is likely to perform better on certain audio files than others. In this section, we try to find which two raw audio waveforms from the 4391 samples in the validation dataset our models performs best and worst on respectively.

The audio sample with the filename *'various_artists-south_by_southwest_compilation-03-completely_mysticism_dr_kuch-0-29.npy'* seems to have the lowest value of binary cross entropy loss at 0.0582 whereas the one with the filename '*lizzi-love_and_you_and_i-02-love_and_you_and_i-233-262.npy'* has the highest loss value of 0.7612. This is a difference of 0.703 between the samples our model performs its best and worst on.

IX. Improvements

A. Increase in Accuracy

To improve the AUC score achieved by the model, alterations to the model architecture along with the addition of appropriate regularization techniques were explored. This improved version proved capable of producing an AUC score of 0.8572 when length and stride provided as input parameters to the model were set at 256 and 256 respectively. This is a staggering improvement of 0.0745 from the AUC score achieved by the base model when passed in the same parameters.

B. Increasing Network Depth

In the improved version of the model, the depth of the neural network was increased from consisting of a total of 7 layers in the base model to now consisting of 9 layers. Increasing the number of hidden layers in the neural network can enable it to learn more hierarchical representations of data. The output of each layer in a network represents its feature space which are all the intricate relationships and patterns it was able to learn from the data that was provided to it as input. In a deeper network, the early layers generally learn simpler low-level features of the data from different parts of the input space. The output of each layer is then delivered as input into the next during forward propagation. This means the deeper layers can be more efficient as they don't have to learn these same features from scratch for different parts of the data anymore, instead, they can build upon existing features already learnt by the previous layers. This facilitates the model in capturing increasingly complex and abstract patterns, which are prevalent is polyphonic music. Additionally, activation functions such as ReLu are used at various layers in the network to transform their output and introduce non-linearity to the model. Adding more layers allows for a more complex composition of these non-linear transformations thereby increasing the flexibility of the model and consequently better equipping it to learn exceedingly complicated features from the data. Overall, increasing the depth of the network can help it learn more sophisticated mappings from inputs to outputs which is also what has been observed in this research.

C. Increasing Network Width

Another change implemented in the base model architecture is increasing the network width. Each convolutional layer in the improved model consists of a substantially greater number of filters. By expanding the width of a layer its capacity to capture a broader range of features can be increased as each filter can now learn to detect a specific feature in the input data. This in return helps the model prevent overfitting to specific patterns in the training data and makes it more robust to variations in the validation data. Increasing the number of filters in hidden layers can also help the network combine features in different ways allowing it to generate more powerful representations of the data. Overall, it helps increase the expressiveness of the model and improves its ability to discriminate between different classes.

D. Regularization

Certain regularization techniques such as batch normalization, dropout, and learning rate schedulers have also been implemented in the improved version of the model. In batch normalization, on a per-batch basis, the inputs to a layer are normalized by subtracting the mean and dividing by the standard deviation of the batch, hence stabilizing the input. These normalized values are then scaled by multiplication with an arbitrary value gamma and then shifted by being added to another arbitrary value beta. Gamma and beta are both learnable parameters which allows the model to optimize scale and shift for each feature during training. This process ensures that the weights within the network don't become imbalanced with extremely low or high values. Not only does this help the improved model achieve faster convergence, but also reduces the ability of outlying large weights to overly influence the training process thereby assisting in improving validation accuracy.

Increasing the depth and width of the network increases the possibility of overfitting to specific features. To tackle this problem, dropout is implemented where each neuron has some pre-defined probability of being temporarily excluded from the forward and backward passes during each training iteration. This prevents any single neuron from being overly reliant of specific features and helps improve the generalizability of the model. Finally, the learning rate scheduler used in this implementation is there to prevent overshooting when the model comes close to convergence and helps continually decrease training loss when validation AUC score becomes stagnated.

E. Model Architecture

The improved model architecture is described in this segment and displayed in figure 5. The first layer is a strided convolutional layer with 128 filters, taking filter length and stride as parameters provided as input to the model. The next six layers consist of three pairs, where each pair is a convolutional layer of filter length 8, filter stride 1 and a max pool layer of pooling size 4, pooling stride 1. However, the number of filters of the convolutional layer in the first pair is 128 whereas the number of filters of the convolutional layers in the following two pairs are 512. The final two layers are still FC dense layers with 100 and 50 units respectively.

All hidden layers use ReLu as an activation function except for the final layer which uses sigmoid. Batch normalization is applied to all four convolutional layers and dropout of 0.5

is applied to the output of the last convolution layer. Learning rate was initially set to 0.01 and decreased by a factor of 5 when the AUC score of the validation data did not increase for more than 3 consecutive epochs and Adam was used as the optimizer. The model was trained for 20 epochs as validated AUC score seemed to plateau after this point.

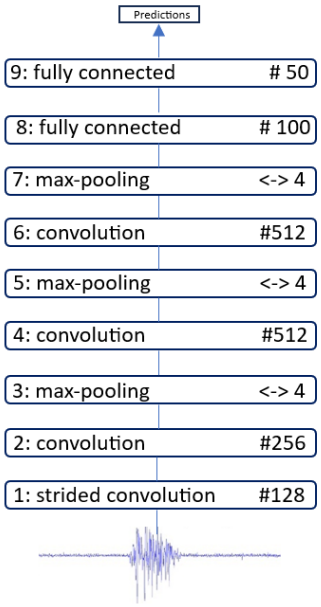

Fig 5. The improved convolutional neural network architecture capable of producing an AUC score of 0.8572. Once again, (#) represents the number of filters in convolutional layers and the number of units in the FC layers. (<->) represents the pooling size of the max pool layers.

F. Training Results when Length and Stride Equals 256

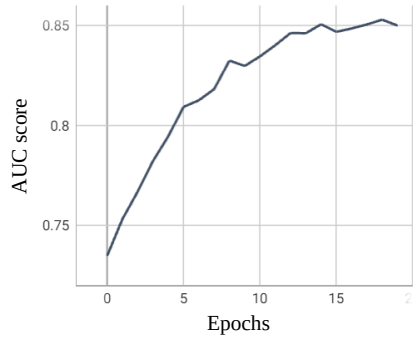

Fig 6. AUC evaluated on validation data every epoch.

Fig 7. Training loss.

G. Other Areas of Exploration

It's worth mentioning that different extents of increasing the depth and width of the network has been explored, however, have eventually been discarded because of their failure to demonstrate any significant growth in accuracy whilst making the model more computationally expensive to train at the same time. Other regularization techniques such as L2 regularization had also been employed, but this seemed to negatively impact the model by reducing accuracy and was thus abandoned. Alternative data pre-processing techniques such as using FFTs to find the first 20 frequencies with the highest amplitudes in the audio waveform, then selecting the highest frequency amongst these 20 and setting all other frequencies higher than this peak frequency to zero, then using the inverse FFT to reconstruct the original waveform has been explored. The hope was to smoothen the waveform and reduce noise since most of the information in an audio signal is generally captured by the lower frequencies. We thought of this to facilitate the model's learning process, however, this too failed to display any significant improvement in accuracy.

X. CONCLUSION AND FUTURE WORK

In this paper, we have managed to replicate the one-dimensional CNN model developed by Dieleman and Schrauwen and achieved adequate accuracy. It has generally been observed that lower values of length and stride provided as input parameters to the model achieves higher AUC scores. Furthermore, we made some architectural changes by increasing the depth and width of the base replica model as well as adding some regularization techniques to further boost accuracy by more than 7% when length and stride inputted is set to 256 and 256 respectively.

Some of the works cited in [1] and [2] provide sample-level inputs to the network and achieve a significantly higher level of accuracy as a result. However, they also use a higher sampling rate and provide 59049 samples to the network which is much greater than the 34950 samples used in this study. They demonstrate that decreasing the stride of the additional strided convolutional layer and increasing the depth of the network has a positive correlation with the AUC scores achieved for audio-tagging. We would like to further explore this idea and measure our model's performance when the input is further granularized and investigate further architectural changes to produce a deeper model attempting to improve accuracy. The use of building blocks inspired from combinations of various existing state-of-the-art classification models have also proven extremely effective in automatic audio-tagging, and this is another area of exploration which can be integrated into the building blocks of our model which currently only consist of a single convolutional and max pooling layer.

REFERENCES

[1] Lee, J., Park, J., Luke, K. and Juhan, K. (n.d.). *SAMPLE-LEVEL DEEP CONVOLUTIONAL NEURAL NETWORKS FOR MUSIC AUTO-TAGGING USING RAW WAVEFORMS.*

[2] ieeexplore.ieee.org. (n.d.). *Sample-Level CNN Architectures for Music Auto-Tagging Using Raw Waveforms | IEEE Conference Publication | IEEE Xplore*.

[3] ieeexplore.ieee.org. (n.d.). *Implementation of Computationally Efficient And Accurate Music Auto Tagging | IEEE Conference Publication | IEEE Xplore*.