Applied Deep Learning Coursework

Nikil Chandrasekar
Department of Computer Science
University of Bristol
student no: 2032447
gv20319@bristol.ac.uk

I. Introduction

The main problem addressed by the paper [1] revolves around music information retrieval tasks and looks to improve on traditional engineered features and shallow architectures using deep architectures and feature learning. Specifically investigating the feasibility of applying feature learning directly to raw audio signals, as opposed to using mid-level representations like spectrograms. The paper aims to reduce the required engineering effort and the need for prior knowledge in music audio processing.

II. Related Work

Following the work of Dieleman [2] which explored end-to-end learning in music audio, the field has seen some notable advancements.

A. Tutorial on Deep Learning for Music Information Retrieval

The paper [3] reviews essential deep learning modules and methodologies, such as dense layers, convolutional layers, recurrent layers, activation functions, and loss functions, all crucial for building effective neural networks for MIR tasks. It emphasizes the versatility of these techniques, initially developed for other domains but now successfully adapted for MIR

III. Dataset

The dataset used in the paper and the one provided to us contained 16 parts in total and was initially provided with a 13-3 split of train and test sets respectively. The provided dataset was then wrangled to include a third validation dataset, finally comprising a 12-1-3 split of train, validation, and test parts respectively. The paper [1] mentions a sample rate of 16 kHz for the audio clips, while the provided dataset contained a 12 kHz sample frequency. This difference in sampling rate could significantly affect the performance of the model and may have contributed to the differences in results produced in my replication and the original paper.

IV. CNN Architecture

The architecture described below served as the base model for replicating the results of the paper for raw audio inputs.

The first layer is a strided convolutional layer of input 1 and output channel of 32 a kernel and stride size of 256, I later experiment with different lengths and strides. The

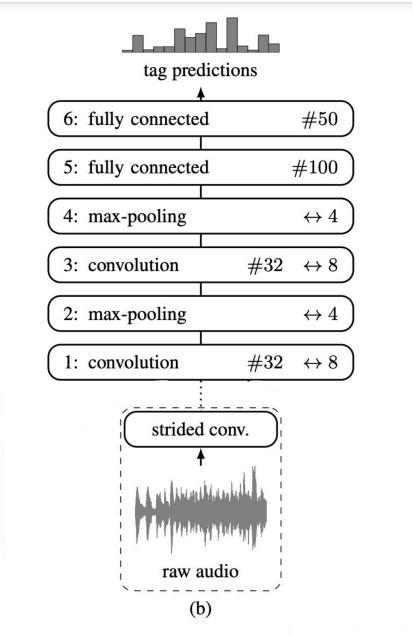

Fig. 1. Architecture Diagram

Strided convolution helps in reducing the computational load, which is particularly important given the high dimensionality of raw audio data.

Following the output from the strided convolution layers, the network architecture consists of two convolutional layers, each with 32 filters of length 8. Following each convolutional layer, there are max-pooling layers with a pooling size of 4.

The network architecture includes two dense (fully connected) layers with 100 and 50 units, respectively at the end, these layers perform the classification task based on the features extracted by the convolutional layers.

Kaiming/He initialization was used to initialize the weights of convolutional and fully connected layers. Rectified linear units (ReLUs) are used in all layers except the top layer. ReLUs help in addressing the vanishing gradient problem, thus aiding in more effective training of deep networks, and a sigmoidal function is used in the final dense layer.

V. Implementation Details

Once the dataset was wrangled to ensure the validation, test, and train sets were organized correctly. the model described in the CNN architecture was implemented. The following steps were crucial in replicating the results of the paper [1].

A. Preprocessing

A critical preprocessing step involved normalizing the audio data. For this, the minimum and maximum values of the audio amplitudes were computed from the training dataset. This range was then used to normalize the data to fall within -1 to 1. Notably, the min-max values derived from the training set were the same as the ones derived from the validation and test, and hence the same were used across all the sets. Other normalization techniques like logarithmic scaling and standard scaling were also tried, these did not seem to provide any significant change in results. Normalization is essential for maintaining consistency in the input data and ensuring that the network trains effectively.

B. Model Training and Evaluation

The training of the model was conducted using minibatch gradient descent. with minibatches containing 10 samples each and the audio input was into windows of 2.9 seconds. This selection captures sufficient data while keeping the computational load manageable.

Each batch was of the size of size [B,10,1,34950] and the first two dimensions were flattened for the batch to work with the 1D convolutional network and reshaped back to its original size

There was also an average pool done across the sub-clips before proceeding to classify the tags

The evaluation of the model's performance was done by computing the area under the Receiver Operating Characteristic (ROC) curve (AUC) for each tag and averaging these across all 50 tags.

The model was also frequently evaluated against the validation set to detect overfitting during training.

C. Tuning and Optimizations

Once the model was running hyperparameters like learning rate, momentum, batch size, and the number of epochs, were selected to be fine-tuned to optimize the model's performance further.

After running the model for 30 epochs, it was clear that the model began to overfit consistently 18th epoch, an optimal epoch size of 20 was chosen. 5 different batch sizes were tested which resulted in no significant difference between them so a batch size of 10 was chosen to run these experiments.

The learning rate and momentum seemed to be key in optimizing the model performance, the combination of learning rates from 0.01-0.2 and momentum values from 0.9-0.99 were vigorously tried and tested and the AUC scores for the following were recorded, which were then used to spot a general trend in the best hyperparameters to use. I concluded that a learning rate of 0.05 and momentum of 0.94 seemed to provide the best results.

VI. Replicating Quantitative Results

Following the Hyperparameter tuning, the combinations of different lengths and strides were tested on the model, the results of which can be seen in the table below.

The incremental improvements in AUC scores with smaller lengths and strides indicate a trend where the network benefits from a higher frequency of feature sampling and increased data overlap. This reveals that for this specific audio classification task, the CNN can leverage detailed local temporal features better than broader ones, leading to more accurate predictions.

Length	Stride	AUC (raw audio)
1024	1024	0.8147
1024	512	0.8130
512	512	0.8250
512	256	0.8267
256	256	0.8285

TABLE I
AUC scores for different configurations of length and stride

VII. Training Curves

Post hyperparameter tuning, the model exhibits a solid learning pattern, as indicated by the increasing accuracy and decreasing loss. However, the slight uptrend in validation loss and the plateau in validation accuracy point to the possibility of overfitting past a certain number of epochs. This could be a cue for early stopping, where training is halted before overfitting becomes more pronounced.

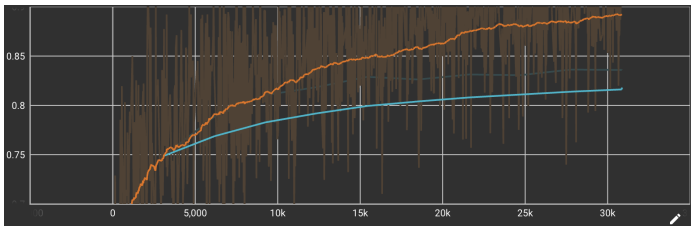

Fig. 2. Accuracy Curves, the blue and orange curves represent the validation and the training sets respectively

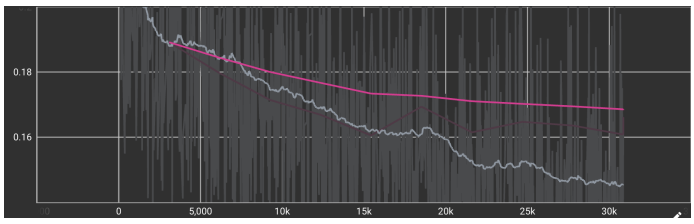

Fig. 3. Loss Curves, the pink and grey curves represent the validation and the training sets respectively

VIII. Qualitative Results

Testing the model against the test set and checking the per-class AUC scores, to evaluate how well my model does on each tag, and the specific file can be found below:

Good example: electric-frankenstein-sick-songs-02-ill-be-standing-on-my-own-117-146.npy
True class: rock
Predicted class: rock

Bad example 1: "beth-quist-shall-we-dance-06-evil-grid-349-378"
True class: no piano
Predicted class: ambient

Bad example 2: "kyiv-chamber-choir-praise-the-lord-13-archangelskyi-think-upon-the-faithful-day-349-378.npy"
True class: no piano
Predicted class: classical

The top-performing classes ('hard', 'rock', 'heavy') as seen in Fig.4 likely possess distinct and consistent audio signatures that are well-represented in the training data, allowing the model to learn clear distinguishing features.

The 'no piano', 'no singer', and 'no beat' classes, defined by the absence of specific musical elements, pose a greater challenge for the model. This points to the possibility that the model is better tuned to recognize the positive presence of features rather than their absence. To improve performance in the lower-scoring classes, strategies such as augmenting the dataset with more examples of these classes or even redefining the classes to be more distinctive could be beneficial.

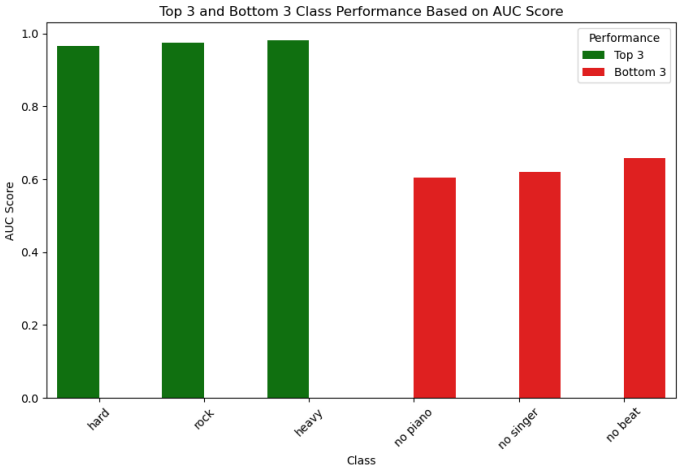

Fig. 4. Performance per class

IX. Improvements

To extend the base model a plethora of options were tried tested and evaluated. In the end, I implemented a spectrogram input version of the model along with batch norm and dropout, the combination of these methods resulted in a significant boost in performance, with a high score of 0.87.

A. Spectograms

The choice to utilize spectrograms as a primary feature representation stems from their ability to provide a comprehensive time-frequency analysis of audio signals. Unlike raw audio, spectrograms translate these signals into a format that visually represents how the spectral density of a signal varies with time. This transformation is crucial for capturing the nuances of music, such as timbre, rhythm, and pitch, making it easier for CNN to detect and classify different music attributes. The SpecCNN model computes spectrograms from the raw audio samples input using signal processing techniques in the forward pass, the output is flattened so the spectrogram inputs still work with the 1D convolutional layer, I also applied dynamic range compression. The spectrograms are not in the mel- range but This approach still facilitates the model's ability to discern intricate patterns in the dataset.

B. Batch Normalization

Batch Normalization is employed at various points in the SpecCNN architecture. It normalizes the input layer by adjusting and scaling the activations, which helps in reducing internal covariate shifts. This is particularly beneficial for training stability, allowing the use of higher learning rates and facilitating faster convergence. In the context of music tagging, where the input data (spectrograms) can have high variability, Batch Normalization ensures a consistent scale across batches, improving the generalization ability of the model.

C. Dropout

Dropout is a regularization technique that has been integrated into the fully connected layers of the model. By randomly setting a portion of the neurons to zero during each training iteration, Dropout prevents complex co-adaptations on training data. This is key in avoiding overfitting, ensuring that the network does not rely too heavily on any individual neuron and, therefore, develops a more robust and generalized understanding of the music features.

The combination of spectrograms, Batch Normalization, and Dropout in the SpecCNN model creates a synergistic effect that enhances overall performance. Spectrograms provide a rich and informative representation of audio suitable for CNNs, Batch Normalization ensures more stable and efficient training, and Dropout enhances the model's ability to generalize to new, unseen data. This integrated approach addresses the complexities and challenges inherent in music tagging, resulting in a more accurate and reliable model capable of discerning a wide range of musical tags.

X. Conclusion and Future Work

A. Conclusion

This report has detailed the process of replicating and extending the original CNN architecture proposed by Dieleman et al. for music tagging, culminating in the development of

Model Type	AUC Scores
Spectrogram-Input + BatchNorm + Dropout	0.86 $\pm$0.01
Spectrogram-Input Only	0.73 $\pm$0.01
Raw-Audio-Input + BatchNorm	0.83 $\pm$0.01
Raw-Audio-Input + Dropout	0.83 $\pm$0.01
Base Model	0.83$\pm$0.01

TABLE II
AUC SCORES FOR DIFFERENT MODELS

the SpecCNN model. This extension was driven by the goal of enhancing the model's ability to accurately classify and tag music based on its audio characteristics.

The introduction of spectrograms as the primary feature representation marked a pivotal shift in the model's approach to handling audio data. By converting raw audio into a time-frequency representation, the SpecCNN model was able to capture the essential elements of music more effectively. This methodological shift acknowledges the complex nature of musical audio and the importance of accurately representing it for effective classification.

The integration of Batch Normalization and Dropout into the SpecCNN architecture further refined the model's performance. Batch Normalization addressed the issue of internal covariate shift, promoting faster and more stable training and Dropout, on the other hand, provided a robust regularization mechanism, reducing the risk of overfitting and enhancing the model's ability to generalize to new data.

B. Future Work

Exploring Convolutional Recurrent Neural Network (CRNN) Architectures One potential avenue is the integration of Convolutional Recurrent Neural Network (CRNN) architectures. CRNNs combine the feature extraction capabilities of CNNs with the sequential data processing strengths of Recurrent Neural Networks (RNNs), making them particularly well-suited for audio analysis. In the context of the SpecCNN model, transitioning to a CRNN could provide a more nuanced understanding of temporal dependencies in music, such as rhythm and melody progression.Investigating CRNNs could uncover deeper insights into the sequential nature of musical elements, potentially leading to more accurate tagging.

Hyperparameter Tuning of the Extended Model Another critical area for future work is the hyperparameter tuning of the SpecCNN model. Given the complexities of audio data and the intricacies of music tagging, fine-tuning hyperparameters like learning rates, dropout rates, and the configuration of neural network layers can significantly impact model performance. Experimenting with different combinations of these parameters could lead to optimizations that enhance the model's accuracy and efficiency.

REFERENCES

[1] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 2014, pp. 6964-6968, doi: 10.1109/ICASSP.2014.6854950

[2] Choi, K., Fazekas, G., Sandler, M. and Cho, K., 2017, March. Convolutional recurrent neural networks for music classification. In 2017 IEEE International conference on acoustics, speech and signal processing (ICASSP) (pp. 2392-2396). IEEE.

[3] Choi, Keunwoo Fazekas, György Cho, Kyunghyun Sandler, Mark. (2017). A Tutorial on Deep Learning for Music Information Retrieval.