Implementation of Shallow and Deep Neural Network Architecture for Music Genre Classification

Tanvi Pandey
Department of Computer Science
University of Bristol
Bristol, United Kingdom
cz19182@bristol.ac.uk

Iban Hossain
Department of Computer Science
University of Bristol
Bristol, United Kingdom
bb19929@bristol.ac.uk

Nisa Bayraktar
Department of Computer Science
University of Bristol
Bristol, United Kingdom
da19157@bristol.ac.uk

***Abstract*—The objective of this report is to design a shallow convolutional neural network (CNN) to classify the music genre of a given dataset. The dataset used for training and testing is GTZAN. Each file is represented as a spectrogram and can be categorized into ten music genres.**

A good reliable shallow CNN has been designed which shows 63.12% accuracy. Along with this an extension of deep CNN has been achieved successfully which shows 62.67% accuracy. Both have been replicated from *A. Schindler et. al.* paper and are trained at 64 batch size. The models have been run for 100 and 200 epochs using BlueCrystal Phase 4 supercomputer. The training and testing accuracy scores have been calculated using the raw accuracy method.

For the second extension, architecture has been proposed after taking inspiration from *K. Choi et al's* paper. The new and unique extension gives 64.52% raw accuracy and is built upon deep CNN.

***Index Terms*—deep learning, music, genre, training, convolutional neural networks, testing, recurrent neural networks**

I. Introduction

Music Information Retrieval (MIR) is a scientific area that focuses on applications such as music genre classification, and music generation by extracting information from the audio. The technologies utilized in MIR include Machine Learning approaches using Artificial Neural Networks (ANNs). Recent research demonstrates that Convolutional Neural Networks (CNNs) successfully perform audio classification. [1]

CNN is a type of ANN that can be categorized as shallow CNN and deep CNN based on the number of convolutional layers. In most Deep Learning applications such as Computer Vision (CV) and Natural Language Processing (NLP), deep network architectures produce better results than shallow networks [2]. However, no significance is found between the shallow and deep CNN architectures supporting this argument in audio classification. [1] The reason behind this problem might be the lack of similar data sample availability in the MIR field to train CNNs.

This report [1] aims to investigate the performance of shallow CNN in genre classification by reproducing the architecture introduced by *A. Schindler et. al.* [1]. The deep CNN architecture in *A. Schindler et. al.* paper is replicated as a first extension which is then improved by applying a Recurrent Neural Network (RNN) layer to the model as a second extension. The details of the shallow CNN architecture and its implementation are explained in sections IV and V. Then, the result of the shallow model is discussed in sections VI, VII, and VIII. Finally, the conclusion and future work are provided in sections X following the IX section where our first and second extensions are explained.

[1]We agree that all members have contributed to this project (both code and report) in an approximately equal manner

II. Related Work

After the publication of the *Schindler's* paper in 2016, music genre classification using neural networks has shown a lot of progress. In 2016 November, *Jordi Pons et al* explored various architectural choices of relevance for music signals classification tasks to start understanding what the chosen networks are learning as CNNs quite often behave like a black box. This paper has demonstrated how convolutional filters with different shapes can fit specific musical concepts. The concepts analyzed by them can be seen as temporal and frequency properties of a dataset consisting of ballroom music audios. They have restricted their design to a shallow network as deeper networks would make the specific filter choices less explainable. The accuracy is computed using 10-fold cross-validation with a randomly generated train-validation-test split of 80%-10%- 10%. Their results indeed show that the musically motivated Time-Frequency architectures can achieve similar results as Black-box deep architecture approaches. [3]

One of the recent papers from 2017 [4], has used a different approach by combining multichannel CNNs with different architectures and LSTM into one unified architecture (Multi-Channel Convolutional LSTM, MCCLSTM) to extract high-level music descriptors. The authors *N. Chen et al* have tested on GTZAN, the Ballroom dataset and the Soundtracks dataset. They have tested 3 different architectures- CNNs, MLP, and LSTMs. The best performing model takes three channels of

CNNs with different shapes of filter and has applied them on each spectrogram to extract the pitch, tempo, and bass relevant labels, respectively. Then concatenation of the outputs of each CNNs channel and passing them through a fully connected layer to obtain the combined label is done. Finally, LSTM is applied on the combined label sequence of the whole track to extract its long-term structure-property to obtain the high-level label. The accuracy of their model on the GTZAN, Ballroom, and Soundtracks datasets is around 84%,91%, and 74% respectively.

Following this, in 2018 a paper by *Dong* addresses the problem of the accuracy of music genre classification still being under 70% that humans could achieve for the same task. However, the paper [5] mentions about using a convolutional deep network resulting in 70% accuracy in a 10-genre classification task. The dataset used by them is GTZAN and has claimed to combine the knowledge from human psychophysics study and neurophysiology called a spectro-temporal receptive field (STRF). They used two convolutional layers to learn features in both frequency and time domains and RELU as the activation function. Dropout and L2 regularisation is used as well in their implementation to prevent extreme weights.

III. Dataset

We used the GTZAN dataset for our classification tasks. [6] This is a collection of 1000 audio tracks that are 30 seconds long and are evenly distributed among 10 different genres: blues, classical, country, disco, hip hop, jazz, reggae, rock, metal, and pop. Each genre is represented by 100 tracks, making the dataset balanced in terms of genre distribution.

The dataset is useful for evaluating the performance of various music classification algorithms and is often used as a benchmark for comparing the performance of different algorithms and models. [7]

The dataset is given in the .pkl format, which is created from the python pickle module. For our model, we used the provided train.pkl and val.pkl files as our training and evaluation data, respectively. These pkl files are lists of tuples with four elements: the filename, the audio spectrogram, the label of the audio file, the audio sample used to create the spectrogram. We specifically use the audio spectrogram and the label part of the tuple to train and evaluate the model. The audio spectrogram is a list that corresponds to the audio segment from a track. The label is an integer from 0-9 that represents the corresponding genre of the audio file, this is used to calculate our loss function and evaluate our model.

There are 1000 audio files and we used a 75:25 train-test split. Each audio file is represented by 15 spectrograms, this resulted in 11250 training samples and 3750 test samples.

IV. CNN Architecture (Schindler et al.)

The architecture for the neural network for the shallow CNN uses a parallel construction of layers and uses typical square filters that can learn the temporal and spectral features at the same time. The architecture comprises two 16-filter kernels followed by a max pooling layer. The padding for the kernels

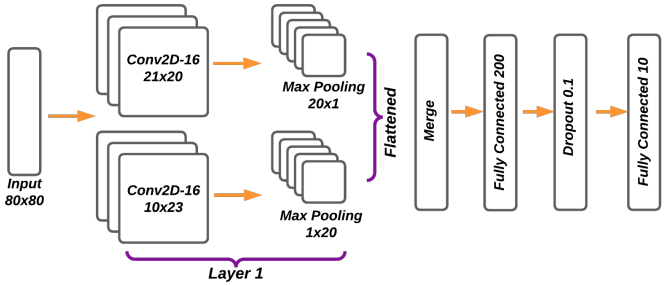

Fig. 1. Shallow CNN

is set to 'same' to support the matching of the outputted feature map to the size of the input of the layer. The left side of the parallel layer constitutes of kernel size 10x23 with the max pooling layer of size 1x20 and is designed to capture the frequency relations. The right side of the parallel layer constitutes of kernel size 21x20 followed by a max pooling layer of size 20x1. This side is intended to capture the temporal features. The pipelines result in an 80x4 vertical rectangular feature map and a 4x80 horizontal rectangular feature map. Both are flattened to a 1x5120 shape and then merged to a 1x10240 shape. This concatenated feature map is then sent through a 200-unit fully connected layer. Before the final step of sending the feature map through a 10-unit fully connected layer is carried out, a dropout of 10% is applied to the 1x200 feature map as shown in Figure 1.

V. Implementation Details

For the implementation of the Shallow CNN, three files are used - `train.py`, `evaluation.py` and `dataset.py`. The creation of the confusion matrix is done in a separate file `visualisation.py`. The `train.py` file has the `Trainer` class and the `CNN` class. The `main` function loads the training and validating dataset from the `dataset.py` file. The model is defined to take 80x80 log-transformed Mel-spectrogram segments as input with one channel. The `criterion` is set as the SoftMax cross-entropy. The optimizer used is `Adam` keeping the values of beta1=0.9, beta2=0.999, epsilon=1e-08, and a learning rate of 0.00005.

The `train` method in `Trainer` gets results from the CNN, the method calculates the L1 regularisation loss by applying a 0.0001 penalty and adds it to the cross-entropy loss for the result obtained. The forward method in CNN implements all the trainable layers with Leaky RELU [8] as the activation function that allows for negative values close to zero to pass through and does not completely cut off activation for negative values. In this implementation, the alpha value is chosen as 0.3. Leaky RELU is also applied to the second last 200 units fully connected layer but not to the 10 units fully connected layer.

The `validate` method in the trainer reads the batch and labels to calculate the prediction. The predictions are sent to the `evaluation.py` This compares the value and displays the raw accuracy. The accuracy is printed after every two

epochs. Here the confusion matrix is updated after every 2 epochs by calling the `visualise` method. The implementation is done in python using the `Pytorch` library. The data was trained on 28, 32, 64, and 128 batch sizes so as to fit our batches entirely in the CPU/GPU memory as they come with storage capacity in powers of 2. Finally, 64 batch size proved to be the most accurate. All layer weight initialization is done using the `Kaiming Normal` [8] with keeping zero bias.

VI. Replicating Quantitative Results

Table I shows our results for the shallow CNN for both 100 and 200 epochs along with our extensions which are discussed in section IX. The raw accuracy is measured by averaging 5 runs. Our shallow CNN accuracy results are consistent with the ones provided in the original paper. When we ran the model for 200 epochs, we observed some minor improvements, which aligns with the findings in the paper.

TABLE I
Raw Accuracy Results

Model	Raw	Epoch
shallow CNN	63.12	100
deep network	62.67	100
CRNN	64.52	100
shallow CNN	63.8	200
deep network	63.1	200
CRNN	64.72	200

Figure 2 is the confusion matrix of our shallow CNN model after 100 epochs. This shows what each genre was classified as, which allows us to analyze where the model performed well and where it could be improved.

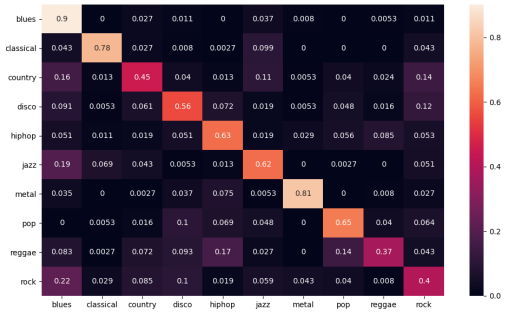

Fig. 2. Confusion matrix for shallow CNN 100 epoch model

The model was quite accurate at correctly classifying genres such as blues, classical, and metal. This could be due to the distinct patterns that these genres have, and our model being able to learn them. The model was not very effective at correctly identifying genres such as country, reggae, and rock. It classified country music as rock or blues which could be due to similar patterns, or the model was simply not able to pick up on the specific features of country music. Jazz and rock were also classified as blues quite a few times, which suggests that the blues genre might have some generic patterns that are present in these other genres.

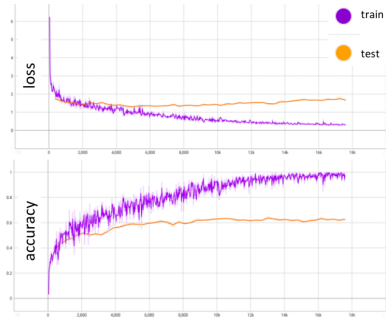

Fig. 3. Accuracy and Loss curves for shallow CNN in 100 epochs

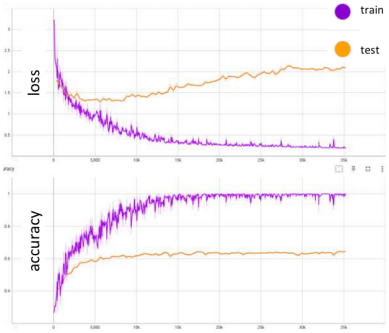

Fig. 4. Accuracy and Loss curves for shallow CNN in 200 epochs

VII. Training Curves

The loss and accuracy curves for both training and test sets are displayed using Tensorboard. Each figure 3,4 represents the exact same logs(runs) mentioned in section VI for 100 and 200 epochs. The training curve results for both epochs clearly demonstrate that our model overfits. Overfitting occurs when the network fits the data perfectly without actually learning it. This means the model memorises all the training data, therefore, it cannot generalise the new data given in the test set. The evidence of this can be seen in the figures 3 and 4 where the test loss curve is going up while training loss is decreasing for both epochs.

VIII. Qualitative Results

The data being passed to the model is represented as Mel spectrograms instead of raw data vectors. A spectrogram represents the change in frequency and amplitude over time for a given signal. By representing the audio samples in the frequency domain we hope to guide the CNN to focus on the types of features that are similar to the human perception of pitch. A representation of the spectrogram also allows us to visually compare different types of genres and analyze how the model is performing.

Figure 5 shows the spectrogram of a sample of rock music, which was incorrectly classified as country music by our model. This is due to the sample having similar features as country music. After analyzing the amplitude and frequency of rock music and country music in general, we found that they both use lower frequencies at higher amplitude. This can be seen by the use of a higher amplitude of around 128 Hz.

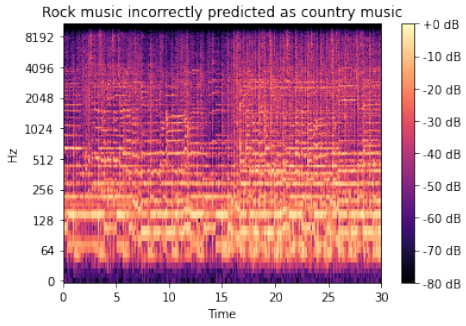

Fig. 5. A spectrogram of rock music incorrectly predicted as country

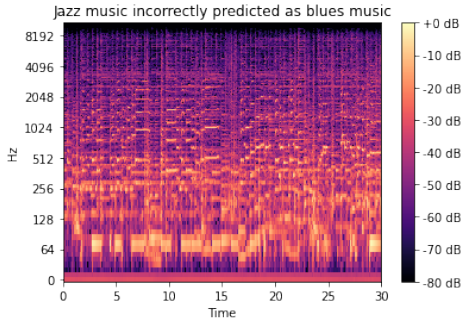

Fig. 6. A spectrogram of jazz music incorrectly predicted as blues

Figure 6 shows the spectrogram of a sample of jazz music that was incorrectly classified as blues music. Having analyzed both jazz and blues music's frequency and amplitude, blues music uses frequencies below 512 Hz at a high amplitude, and a frequency higher than this at a low amplitude. Jazz music also shows a similar pattern, although blues music uses more constant low frequencies. The model was not able to distinguish between the two in this instance which resulted in the incorrect prediction.

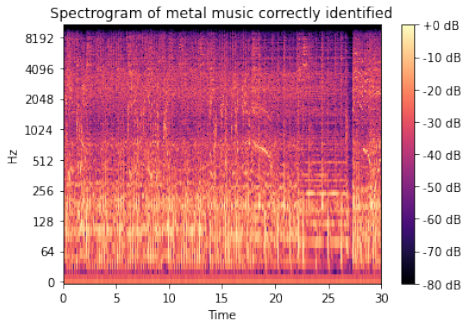

Fig. 7. A spectrogram of pop music correctly identified

Figure 7 shows a correctly classified sample of metal music from our model. Analyzing the frequency and amplitude of metal music showed that it has relatively fewer changes in the amplitude of certain frequencies over time. Frequencies around 128 Hz were used with higher amplitude, as can be seen in the figure. Overall metal music used frequencies with a strong emphasis on the lower end of the spectrum, which might correspond to the guitar riffs of the genre. The amplitude in these frequencies was also high which can be associated with the genre. These patterns were likely picked up by our model, which correctly identified metal music 81% of the time as shown in the confusion matrix in figure 2.

IX. Improvements

A. First Extension

1) ***Deep Network Architecture****:* For our first extension, the deep network architecture in the original paper was replicated to see the accuracy difference between the shallow and deep networks. In the deep architecture 8, 3 more convolutional layers were added to the shallow network in the same pipeline structure explained in section V. The dropout value was changed to 0.25 as mentioned in the paper and the full implementation was made in `deepTrain.py` script. Both pipelines have kernel sizes getting halved as they pass through each layer. The left pipeline comprises 10x23, 5x11, 3x5, and 2x4 kernels, each followed by a 2x2 max pooling layer, except the last which has 1x5 max pooling. The right parallel pipeline consists of 21x20, 10x5, 5x3, and 4x2 kernels, again followed by a 2x2 max pooling layer. The last layer however has 5x1 max pooling. This was done to have the same shapes with the result of the right pipeline rotated by 90 degrees.

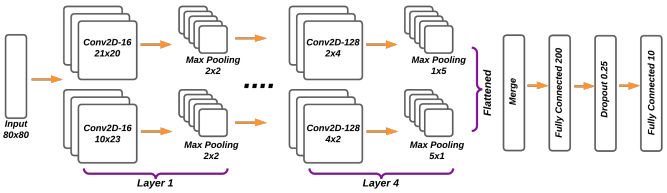

Fig. 8. Deep CNN architecture

2) ***Deep Network Results****:* The raw accuracy results of our deep network for 100 and 200 epochs are identical to the original paper's results where the deep network has no significant difference from the shallow network. As seen in table I, the shallow network even performs better than the deep network for both epochs. In 200 epochs the deep network does not experience a significant increase compared to 100 epochs, unlike the shallow network. This might be because the over-fitting in the deep architecture occurs earlier than in the shallow network as mentioned in the paper [1].

B. Second Extension

To improve the deep architecture as a second extension, detailed research about recent trends in music classification was made. The papers that were found, claim that adding a Recurrent Neural Network (RNN) layer to the convolutional network increases the accuracy [9], [10], [11], [12]. This architecture is called a Convolutional Recurrent Neural Network (CRNN) where convolutional layers and RNN layer are combined.

*1) **RNNs**:* RNNs contain a recurrent layer that allows cyclic training in the network by saving the output of the layer to be reused as an input which is useful for sequential data. Therefore, using RNNs gives better results than CNNs on text, videos, and audio data. [13] This is because CNNs pass the data between their layers simultaneously therefore, they are more suitable for spatial data. Current researches suggest that using Long-Short Term Memory (LSTM) is the state of art for training wave format data and might be the solution to MIR problems [13], [9], [10]. LSTM is a gated RNN type which contains gates additional to the recurrent layer that help the network to keep only the beneficial information. However, RNNs and CNNs still have limitations when they are individually used for music classification [13]. For example, CNN's convolutional layers are beneficial for training the local context of the spectrogram such as the instrument type but unlike RNNs, it cannot deal with long-term dependencies such as the rhythm of the music [13].

*2) **CRNNs**:* CRNNs were introduced as a solution to this problem and tested by some researchers on audio classification tasks [12], [4], [11]. The papers designed a new architecture called CNN_LSTM or in general CRNN which is a combination of RNN and CNN. This model utilizes convolutional layers from CNN and applies LSTM layers to the network. To implement this architecture as our second extension, the papers [11], [10], [4] were followed because the architecture of our deep network was similar to the architectures in these papers and they tested their models using the GTZAN dataset.

*3) **CRNN architecture**:* Our CRNN is almost identical to our original deep architecture as shown in Figure 9. The changes that we made were replacing the first fully connected layer with an LSTM layer using `PyTorch's` functional LSTM `nn.LSTM()`. Then, apply 0.25 dropout just in the LSTM layer to prevent overfitting as described in paper [11].

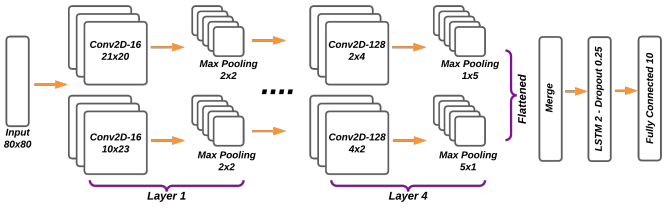

Fig. 9. CRNN architecture

*4) **Results of CRNN**:* We trained our CRNN for 100 and 200 epochs. In the accuracy table I it can be seen that the results of our CRNN were promising where we managed to improve our deep CNN's accuracy by almost 2%. The 200 epoch performed better which we also experienced in our shallow and deep architectures. Our CRNN experienced the same trend from the papers [11], [4], [10] and verified CRNN's better performance for music classification.

X. Conclusion and Future Work

In this report, the shallow architecture from *A. Schindler*'s paper has been replicated to classify the music genres in the GTZAN dataset. The replicated network was almost identical to the original paper, except for some missing information, such as hyper-parameter values and fully connected layers mentioned in section V. The deep network in the paper was reproduced as a first extension to investigate the performance change between shallow and deep network architectures. The results obtained for both networks have the same trends as described in the original paper, where implementing more convolutional layers to the shallow network did not improve the accuracy score [1]. To improve the deep network performance, the LSTM layer was implemented which was inspired by CRNN model introduced in the papers [11], [4], [10]. Applying this second extension improved our deep network results by almost 2% reaching around 64.5% accuracy.

Future work can be implementing Bidirectional LSTM (BiLSTM) to increase the performance of the network even more by providing additional training of the data using multidirectional gates as explained in the research [13]. Another area for future work would focus on the non-deterministic behaviour problem of RNNs in CUDA that we also experienced while training our CRNN. This is a common problem caused by CUDA's RNN functions as it was warned on their website [8]. Therefore, we set our seed in the code to reproduce the accuracy results as suggested in the `PyTorch` website for this problem [8]. Overall, being able to reproduce the architectures in *A. Schindler et. al.*'s paper, and improving the deep architecture was a success for this project.

References

[1] A. Schindler, T. Lidy, and A. Rauber, "Comparing shallow versus deep neural network architectures for automatic music genre classification." *FMT*, pp. 17–21, 2016.

[2] K. Choi, G. Fazekas, K. Cho, and M. Sandler, "A tutorial on deep learning for music information retrieval," *arXiv preprint arXiv:1709.04396*, 2017.

[3] X. S. J. Pons, T. Lidy, "Experimenting with musically motivated convolutional neural networks." *14th International Workshop on Content-Based Multimedia Indexing (CBMI) (2016)*, 2016.

[4] N. Chen and S. Wang, "High-level music descriptor extraction algorithm based on combination of multi-channel cnns and lstm," *Proceedings of the 18th ISMIR Conference, Suzhou, China*, October 2017.

[5] M. Dong, "Convolutional neural network achieves human level accuracy in music genre classification," *arXiv:1802.09697*, 2018.

[6] G. Tzanetakis, "Manipulation, analysis and retrieval systems for audio signals," Ph.D. dissertation, Princeton University, 2002.

[7] B. L. Sturm, "The gtzan dataset: Its contents, its faults, their effects on evaluation, and its future use," *Journal of New Music Research*, 2014.

[8] PyTorch, "Pytorch.org," https://pytorch.org/.

[9] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 4580–4584.

[10] Y.-H. Cheng, P.-C. Chang, D.-M. Nguyen, and C.-N. Kuo, "Automatic music genre classification based on crnn." *Engineering Letters*, vol. 29, no. 1, 2020.

[11] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *2017 IEEE International conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2017, pp. 2392–2396.

[12] D. Dalmazzo and R. Ramirez, "Mel-spectrogram analysis to identify patterns in musical gestures: a deep learning approach," *MML 2020*, p. 1, 2020.

[13] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "The performance of lstm and bilstm in forecasting time series," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 3285–3292.