Comparing shallow versus deep neural network architectures for automatic music genre classification

Luke Storry
Computer Science
University of Bristol
ls14172@bristol.ac.uk

Sydney Dimmock
Computer Science
University of Bristol
sd14814@bristol.ac.uk

Matthew Co
Computer Science
University of Bristol
mc15184@bristol.ac.uk

I. Introduction

Tasks within Musical Information Retrieval (MIR) have typically relied on hand crafted feature extraction methods before the application of machine learning methods, the construction of which usually rely on strong domain knowledge within the fields of signal processing and musical theory. The current success of Convolution Neural Networks (CNN's) largely motivated by their powerful feature learning properties have shown that this may no longer be necessary as these networks should be able to discover the optimal features of the dataset automatically.

CNN's are broadly split into two architectures, *deep* networks and *shallow* networks. Musical classification tasks that have currently been implemented within deep architectures have not shown to yield similar results to relative tasks performed within image classification [4]. One reason for this may be due to the discrepancy of similarly large datasets that allow for proper training of a deep networks. The aim of this report is therefore to demonstrate a high performance shallow architecture to be used within music classification tasks. Throughout, the comparative performance of deep networks will also be given to encourage theoretical discussion.

II. Related Work

One of the first papers attempting to address the problem of Musical Genre Classification was the 2002 paper by Tzanetakis and Cook [3]. They used three feature sets for representing timbral texture, rhythmic content and pitch content. Statistical pattern recognition classifiers were trained using real-world audio collections to achieve 61% accuracy for ten musical genres. They point out that although their results are far better than chance, and a big step forward in genre classification, humans have been shown [10] to be able to judge genre with an accuracy of 70%.

The paper by Dong [5] attempts to address the problem of achieving human-level Music Genre Classification accuracy through the use of CNNs. They achieved this by using two convolution layers to replicate the human auditory system which was found to work in a hierarchical way. By doing this, the CNN was encouraged to learn features in both the frequency and time domains. The authors highlight that their results demonstrate accuracy very similar to humans due to some of the learned filters being very similar to textitspectro-temporal receptive fields (STRF) a property that can be found within neurons in the human auditory system.

The paper by Jordi Pons et al [6] attempts to address the discrepancy between handcrafted features within Music Information Retrieval and those learned by a 'black box' CNN architecture.

In their paper they demonstrate musically motivated filter choices that encourage networks to learn a particular representation, these choices can be thought of as general musical concepts such as temporal/rhythmic and frequency/pitch properties that will be concretely realized by the network for a given training data set. By doing so they inherently restrict the design to a shallow architecture, this is because for a deeper network specific filter choices become *less* interpretable making specific behavior harder to encourage. The results given by this paper indeed support this claim, a shallow network with musically motivated filter sizes and pooling windows achieves the same performance of the more expressive 'black box' deep architecture but with a fraction of the parameters. The deep architecture consisted of $3,275,312$ parameters, whereas the shallow network gave the same result with $196,816$. Their network architecture blueprint can be seen in Figure 1, vertical filters are aimed at capturing spectral features whilst horizontal filters are argued to be more suitable for learning temporal cues when applied to spectrogram inputs.

III. Dataset

The dataset used for classification was GTZAN [2]. The GTZAN dataset is the most-used public dataset for evaluation in machine listening research for music genre recognition [11], and appears in at least 100 published works.

This dataset consists of 1000 audio tracks that are each 30 seconds long, and are categorized into 10 genres (resulting in 100 tracks per genre). All tracks are 16-bit audio files in the `.wav` format, recorded with a sampling rate of 22050 Hz.

The formulation of this dataset for use within a programming environment is as a dictionary of values split into three

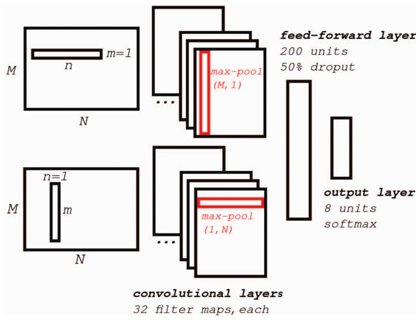

Fig. 1. Shallow architecture proposed by Pons et al [6]

separate fields. The 'labels' field is a list of `int64` values that represent the observed genre of music for a given track sample. The 'data' field is a list of type `numpy.array`, where each of these arrays correspond to a single audio segment taken from a given track (each track has 15 randomly chosen segments). The final entry is under 'track_id', this is a list of integers that identify the track that a given audio segment belongs to. The format of the dataset is `.pkl` created from pythons `pickle` module.

We used the given 750:250 training-test split, resulting in 11250 training segments and 3750 test segments.

IV. Spectrogram Introduction

Instead of performing classification on the raw data vectors, we first represent the samples as *mel* spectrograms. These spectrograms generated from the audio samples are representations of the sample in the frequency domain. Specifically they encode how the amplitude and frequency vary over time for the signal under consideration.

The mel scale is another way of representing the frequency spectrum in a form that tries to capture the differences in frequency according to human sensory perception. This scale scales roughly linearly with frequency up to 1000Hz after which there becomes an exponential difference between the frequency value that gives another step of 1000 in the mel scale. An interpretation of this is that sensory perception of same size pitch intervals at higher frequencies become harder to distinguish by ear. This can be observed by inspecting the relationship between the two scales from the formula given below [1]. By using this mel scale representation of the frequency domain as input to a CNN we are making assumptions that the CNN will learn features of the input signal based on the way it is assumed humans perceive differences in pitch, rather than letting the CNN decide features based on raw input alone.

$$mel(f) = 2595 \cdot log_{10}\left(1 + \frac{f}{700}\right) \tag{1}$$

and its inverse, where $c = 2595/ln(10)$.

$$freq(m) = \left(\exp\frac{m}{c} - 1\right) \cdot 700 \tag{2}$$

The method of generating a spectrogram is to first discretise the given signal into time bins, then calculate the

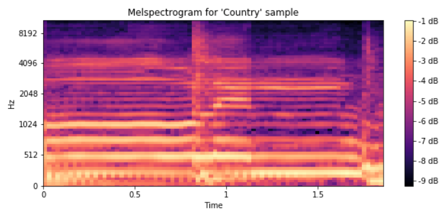

Fig. 2. Mel spectrogram for a Country sample from GTZAN.

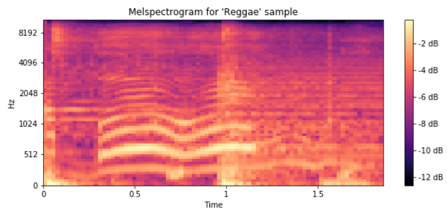

Fig. 3. Mel spectrogram for a Reggae sample from GTZAN.

frequency and amplitude for the given segment using the Fourier transform. Some examples of spectrograms generated from the GTZAN training set are given in Figure 2 and 3. This representation clearly allows for better analysis of the signal properties as they can be visualised, for example in Figure 3 we can notice that the frequencies between 0 and 1024Hz harmonize between a given time segment shown by the wave like forms.

V. Method (Schindler et al)

Schindler et al [4] showed that for smaller data sets, shallow networks are more suited to performing music classification tasks unless a lot of data is available. However, they also showed that corresponding deep networks show significant performance increases when used in conjunction with data augmentation methods.

The architecture of the networks used for both shallow and deep networks adopted a parallel construction of layers, where one pipeline would focus on spectral feature extraction in the *frequency* domain, and the other would identify *temporal* features. This design was motivated from the work performed by Jordi Pons et al [6]. Where $m < M$ by 1 vertical rectangular pooling was used to propagate temporal information, and 1 by $n < N$ horizontal rectangular pooling was used for propagating spectral information in the network. However, the implementation by Schindler et al [4] does not opt for $M = m$ by 1 filters for learning frequency features, and similarly 1 by $N = n$ filters for learning temporal features as originally proposed by pons [6]. Instead they use more typical square filters that can both learn temporal and spectral features at the same time but are harder to motivate and reason about.

VI. Implementation Details

We approached the implementation description provided in the original paper [4] in a modular fashion: each network was specified in its own file, `deepnn.py` and `shallownn.py`,

Model	raw	max	maj	ep
shallow	0.652	0.812	0.788	100
shallow	0.653	0.804	0.784	200
deep	**0.691**	**0.848**	**0.828**	100
deep	0.678	0.832	0.828	200

TABLE I
BASELINE RESULTS FOR DEEP AND SHALLOW NETWORKS USING THE THREE PREDICTION SCORES.

helper functions for data, such as importing, augmentation and the creation of the mel-spectrograms were placed seperately in a `data.py` file. The Tensorflow graph can be built, trained and tested by running `main.py` with a variety of command line options.

After importing and initializing data, we perform the augmentations mentioned in [4] before training is started. This consisted of randomly choosing 3 segments per audio track, applying four different time stretches to each segment, and then applying four different pitch shifts to each of those time-stretched samples. This resulted in 48 new audio samples per track.

When building the Tensorflow graph, extra steps were taken to ensure that the data was pipelined efficiently, as to prevent the bottlenecks encountered in our first prototype. Originally we were splitting the data into batches of 16 outside of Tensorflow's graph then feeding it in, this resulted in very slow training times due to data delivery issues. However, by switching to using Tensorflow's `Dataset` class to deliver data for the next step before the current step had finished, less time was wasted waiting for data to be loaded onto the GPU, allowing for much more efficient implementation. This decreased the time taken to train for 100 epochs from 10 hours to under 10 minutes.

The deep and shallow networks were defined using the API provided by tensorflows `tf.layers` module. Each convolution was specified to use *same* padding with a stride of 1. Each max pooling operation specified strides equal to the pooling window size. All parameter initialization was done though Xavier initialization [7].

Training steps were performed with the Adam optimizer, using a batch size of 16, following this validation was performed once per epoch.

VII. Replicating Quantitative Results

Our results without using data augmentation based on the architectures proposed by Schindler et al [4] can been seen in Table I, where the best value attained for a given metric is emphasized in bold. Overall they are *consisent* with the ones provided in the original paper, and confirm the minor differences in performance on this task between shallow and deep networks as discovered by Schindler et al [4].

Our results do not replicate the improvement observed in *max* and *raw* measurements when training the shallow network for longer, instead we notice a very small decrease. The effect of training the deep network for longer is however the same; we observe no significant performance increase due to overfitting very early in the training process [4]. This can also be seen in Figure 4 and Figure 5.

VIII. Training curves

This section presents graphs of the networks over learning, featuring accuracy and loss metrics for both training and validation steps. These graphs clearly show that overfitting is occurring in both network architectures based on the simultaneously increasing loss function on validation steps, and decreasing loss function for training steps. The interpretation of this is that the networks are fitting the training *too* well, and are becoming poor at *generalising*. Figure 6 gives a good example of this, notice this is particularly acute for the deep network and is due to the larger degree of freedom present in the classification boundary within the input space compared to that of the shallow architecture, allowing for overfitting to occur more easily.

We can also notice from Figure 5 and Figure 7 that validation accuracy can still increase even though validation loss is increasing for the deep network which was also noticed by Schindler et al [4]. This does not appear to be the case for the shallow network where accuracy tends to decrease on longer training times, and is backed up by the results in Table I.

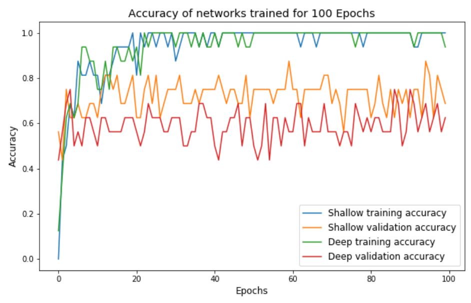

Fig. 4. Training and validation curves depicting accuracy for shallow and deep networks trained for 100 epochs.

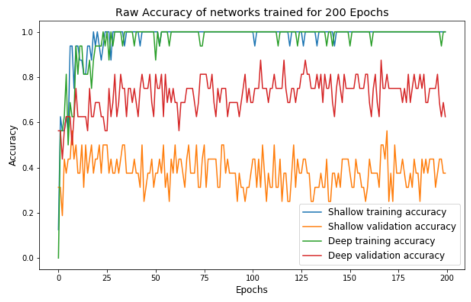

Fig. 5. Training and validation curves depicting accuracy for shallow and deep networks trained for 200 epochs.

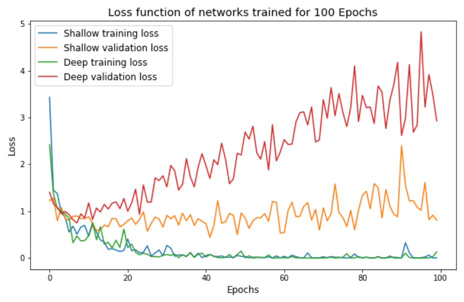

Fig. 6. Training and validation curves depicting loss for shallow and deep networks trained for 100 epochs.

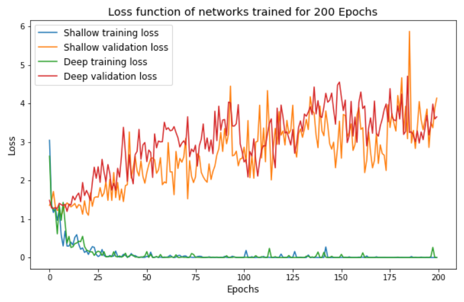

Fig. 7. Training and validation curves depicting loss for shallow and deep networks trained for 200 epochs.

IX. Qualitative Results

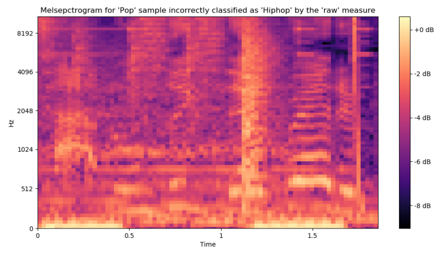

Fig. 8. Mel spectrogram for a Pop sample incorrectly classified as Hiphop by the 'raw' measure from GTZAN.

Figure 8 shows the 'raw' measure incorrectly classifying a Pop track as a Hip-Hop one, the track is correctly classified by max probability and majority voting. It incorrectly classified the sample since more features that commonly occur in Hip-Hop samples were identified than those found in Pop. When our algorithm correctly classified all samples, we found that Pop generally uses a range of frequencies without changing over short periods of time, with a few of those having high amplitude. In contrast, hip-hop has more changes in frequency over short time periods and uses a specific number of frequencies concurrently. They both make use of high frequencies with low amplitude. With this considered, Figure 8 shows that the spectrogram has hip-hop-like features at around 1.5 seconds, where specific frequencies are used rather than a range. In addition, the frequency changes within short time periods instead of having constant frequency. Between 0 and 1 seconds, the spectrogram has more pop-like features, such as frequencies that do not change over short time periods. High frequencies at low amplitudes are used which contribute to the spectrogram being both Pop and Hip-Hop.

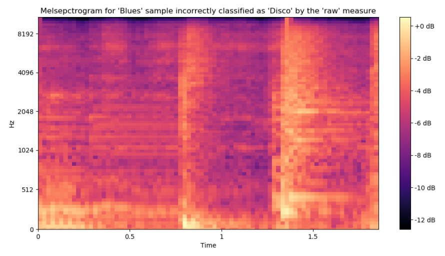

Fig. 9. Mel spectrogram for a Blues sample incorrectly classified as Disco by the 'raw' measure from GTZAN.

Figure 9 shows the 'raw' measure incorrectly classifying a Blues track as a Disco one where Blues is instead identified as the second class prediction. It incorrectly classified the sample since more features that commonly occur in Disco samples were identified than those found in Blues. When our algorithm classified all samples, we found that Blues generally uses constant low frequencies below 512 Hz at a high amplitude and uses a large range of constant frequencies ranging from 512 to 8192 Hz at a low amplitude. In contrast, Disco uses constant frequencies ranging from 0 to 1024 Hz at a high amplitude, has short periods of time where all frequencies are used and uses higher frequencies for short periods of time at a low amplitude. With this considered, Figure 9 shows that the spectrogram has Disco-like features at around 0.75 and 1.25 seconds, where there is a short period of time that all frequencies are used. In addition to this, while there are higher frequencies that are at low amplitude which is a feature consistent with both Blues and Disco, these frequencies are used for short periods of time before stopping and then used again which supports a classification of Disco.

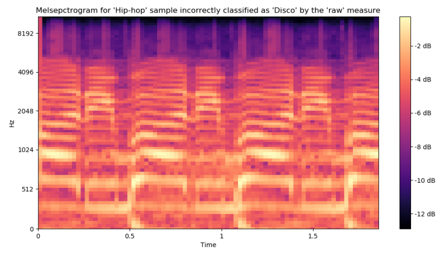

Fig. 10. Mel spectrogram for a Hip-Hop sample incorrectly classified as Disco by the 'raw' measure from GTZAN.

Figure 10 shows the 'raw' measure incorrectly classifying a Hip-Hop track as a Disco one where Hip-Hop is identified as the last class prediction. It incorrectly classified the sample since none or few features of Hip-Hop were recognised by our algorithm. As stated before, Hip-Hop has changes in frequency over short time periods, uses a specific number of frequencies concurrently and make use of high frequencies with low amplitude. With this considered, when looking at Figure 10 the spectrogram seems to have Hip-Hop features with the use of selected frequencies that change over short periods of time, however these frequencies are quite close to each other and also use the frequencies in between each other at a lower amplitude. The spectrogram also does not show use of really high frequencies except at the very beginning at 0 seconds. Because of these, our algorithm was not be able to detect any Hip-Hop features for this sample.

X. Improvements

Model	raw	max	maj	ep
shallow aug	0.637	0.800	0.812	100
shallow aug	0.638	0.808	0.808	200
shallow bn	0.679	0.800	0.788	100
shallow bn	0.695	**0.864**	0.828	200
shallow bn+aug	0.664	0.808	0.820	100
shallow bn+aug	0.663	0.832	0.820	200
deep aug	0.674	0.832	0.824	100
deep aug	0.691	0.840	**0.844**	200
deep bn	**0.713**	0.824	0.824	100
deep bn	0.690	0.832	0.800	200
deep bn+aug	0.674	0.840	0.816	100
deep bn+aug	0.690	0.832	0.812	200

TABLE II
RESULTS FOR DEEP AND SHALLOW NETWORKS WITH BATCH NORMALISATION AND DATA AUGMENTATION USING THE THREE PREDICTION SCORES.

We implemented the data augmentation improvement explained in the original paper [4], alongside Batch normalisation [8].

The justification for using data augmentation is that it attempts to address the comparatively overall lack of data present to train networks for MIR when compared to other typically well performed tasks by CNN's such as image classification. It also allows networks to be more robust when presented with unseen data, providing better *generalisation*, for example the data augmentation we used both includes a pitch shift, and a tempo shift. These deformations are justifiable since if you speed up a song, the genre remains the same, equally if you pitch shift it, while it may sound different the genre has not inherently changed. Data augmentation is also particularly suited to improving the performance of deeper networks, as the larger number of network parameters requires us to train with more data. Table II shows similar performance gains when using data augmentation with deep networks as those proposed by Schindler et al [4]. The results also replicate the decrease in performance of shallow networks when data augmentation is used.

Our theoretical justification for the use of Batch normalisation in the music classification task is based on the reduction of covariate shift that it provides. This regularisation of the distribution of inputs (song tracks) that should be classified by the same ground truth function should speed up training, allowing better results to be obtained from the networks. Our assumption here is that this extra noise over inputs with the same ground truth is generated by the musicians personal style, as acoustic instruments will inherently capture *how* they are played, even if they are used within the same genre.

This assumption is justified by the results in Table II. We can see that the best overall results for Deep and Shallow networks in the 'raw' and 'max' measures respectively have been obtained with this improvement. An interesting result is that the combination of both methods when training a network does not tend to give better performance over using one of the improvements individually.

XI. Conclusion and Future Work

This report has implemented, reviewed and evaluated the performance of deep and shallow networks applied to the music classification task proposed by Schindler et al [4]. Overall we have obtained consistent results with those cited in the original paper, minor differences may have resulted due to some implementation details not described in the original, such as parameter initialisation and batch size.

We have shown that shallow networks give performance very close to deep networks especially in the absence of data augmentation techniques. In addition to this, we have also shown that batch normalisation should be considered when attempting to improve the classification performance of both network architectures.

Future work would focus more on rigorous hyperparameter selection, especially alongside the implementation of batch normalisation proposed, as there is evidence that it makes the network more robust against hyperparameter changes [9]. Being able to obtain optimal hyperparameters for the deep and shallow architectures that can also be theoretically justified would greatly help in terms of gaining a better understanding of subtleties of this classification task and of the network architectures as a whole.

References

[1] Douglas O'Shaughnessy (1987). Speech communication: human and machine. Addison-Wesley. p. 150. ISBN 978-0-201-16520-3.

[2] G. Tzanetakis. Manipulation, analysis and retrieval systems for audio signals. PhD thesis, 2002.

[3] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. IEEE, 2002,

[4] Schindler, Alexander et al. Comparing Shallow versus Deep Neural Network Architectures for Automatic Music Genre Classification. (2016)

[5] Dong, M., 2018. Convolutional Neural Network Achieves Human-level Accuracy in Music Genre Classification. arXiv preprint arXiv:1802.09697.

[6] Pons, Jordi et al. Experimenting with musically motivated convolutional neural networks. 2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI) (2016): 1-6.

[7] Xavier Glorot, Yoshua Bengio ; Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, PMLR 9:249-256, 2010

[8] Ioffe, Sergey and Szegedy, Christian. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift (2015).

[9] Santurkar, Shibani and Tsipras, Dimitris and Ilyas, Andrew and Madry, Aleksander. "How Does Batch Normalization Help Optimization?" (2018).

[10] D. Perrot and R. Gjerdigen, Scanning the dial: An exploration of factors in identification of musical style, in Proc. Soc. Music Perception Cognition, 1999.

[11] Bob L. Sturm The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use CoRR, 2013.