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

Louis Wyborn
Department of Computer Science
University of Bristol
Bristol, United Kingdom
lw15771@bristol.ac.uk

Ahmer Butt
Department of Computer Science
University of Bristol
Bristol, United Kingdom
ab15015@bristol.ac.uk

Stefan Klaus
Department of Computer Science
University of Bristol
Bristol, United Kingdom
sk15962@bristol.ac.uk

***Abstract*—In this report we examine the paper by A. Schindler, T. Lidy and A. Rauber. Firstly detailing similar works and their relation to the paper, then re-implementing the architecture described in the paper in order to reproduce the results found. We then propose 2 further adaptations and show how these improve upon the results found.**

I. Introduction

Audio classification is a well-researched problem in computer science, with many different approaches. Current state-of-the-art methods commonly include two main steps; first hand-crafted feature extraction techniques are used to attach semantically meaningful labels to audio samples, then a machine learning algorithm is used to discriminate between classes. The first step in particular often requires extensive domain knowledge in music or acoustic theory. This has inspired the use of Deep Neural Networks (DNNs) in this field, as they possess the ability to learn these features which previously had to be manually crafted. Convolutional Neural Networks (CNNs) have had much recent success with image classification, as their grid structure aligns well with the structure of images. Even more recently CNNs have been applied to audio classification, through pre-processing the audio data into image-like representations.

Deeper networks have been shown to be able to better represent non-linear relationships of input data than shallow networks in image classification tasks, however this same advantage has not been demonstrated in audio classification tasks. The provided paper therefore addresses this, focusing on comparing the performance of deep networks against shallow ones on different sized datasets on the task of classifying music by genre.

II. Related Work

The provided paper builds on an earlier paper published by the same authors. In the paper ”CQT-based convolutional neural networks for audio scene classification” [1] the authors introduce 2 key ideas; namely using parallel pipelines in the architecture of the neural networks, and using a Constant-Q transformed (CQT) input. This transforms the input into the frequency domain in such a way that lower frequency signals have greater spectral resolution, and higher frequency signals have greater temporal resolution. The paper utilises parallel pipelines due to the semantic differences between each axis of a CQT spectrogram. With image data both axes represent the same semantic meaning, i.e. spatial displacement, however with in this case frequency is along one axis and time along another. Therefore, this paper introduced long, narrow kernel sizes, with one pipeline positioning its kernels aligned along one axis, and the other along the other axis, to capture frequency relations in one pipeline and temporal relations in the other.

The task varied from that of the provided paper, in that it was to classify audio samples by the environment where they are recorded, such as in a café, library, grocery store, etc. They achieved a 10.7% improvement over the baseline provided by the dataset authors.

The paper titled ”Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks” [2] was published after the provided paper, in 2017, however provides justification for representing audio input data into mel-scaled spectrograms over other representations when working with environmental sound data. This paper compared applying CNNs to linearly-scaled spectrograms, mel-scaled spectrograms, CQT spectrograms, and continuous Wavelet transformations (CWTs). They also compared these results against baseline hand-crafted Mel-frequency cepstral coefficients (MFCCs) combined with Guassian Mixture Model (GMM) class separation. The task was the same as the previous paper, to perform audio scene classification. They found that CNNs applied to mel-spectrograms achieved the highest accuracy on both datasets tested, marginally winning out over CQT in most cases. The paper also posits that ”CQT may be even more beneficial for music analysis where timbre signatures are more evident in the instrumentation as compared to environmental sounds”. We will follow up on this later.

The paper titled ”Rectifier Nonlinearities Improve Neural Network Acoustic Models” [3] published in 2013 provides the experimental backing behind the use of rectified linear activation functions over sigmoidal activation functions, specifically with speech recognition tasks. The paper experiments with tanh, ReLU, and LeakyReLU activation functions, with 2-, 3-, and 4-layer CNNs. They found that ReLU activation functions significantly improved accuracy in all different depth network over tanh, likely due to the lack of vanishing gradients. However, LeakyReLU offered no further improvement in accuracy over ReLU. The paper showed ReLU functions gave a 2% absolute reduction in word error rates in continuous speech recognition over tanh activation functions, and a 6% absolute reduction over GMM alternatives to DNNs.

III. Datasets

We used the GZTAN dataset in this project. This is a dataset of 1000 tracks evenly spread across 10 categories: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock. The pickled dataset provided is a version of the GZTAN dataset where each track has been split into 15 randomly chosen 0.97 second long segments, with each segment having an associated label and track id. There is a training/test split of 750/250 tracks, which corresponds to 11250/3750 samples. The dataset provided is formatted as two dictionaries, one test and one training, each with 3 lists; a list of samples, a list of category labels for each sample, and a list of track IDs for each sample. The track IDs of two samples match if they are sampled from the same track. The samples are all in the .wav file format.

IV. Spectrogram Introduction

CNNs work with constant-size grid-structured input data, however audio recordings are not commonly structured as grids, they are 1-dimensional arbitrary-length arrays. Therefore, we need some method of transforming the audio input into a representation CNNs can work with.

Spectrograms are a visual representation of sound recordings. They are a graph with time along the X axis and frequency along the Y axis. The amplitude of the frequencies is represented as a heatmap across the graph.

In the paper the spectrogram is scaled according to the log-mel scale. This is a perceptual scale of pitches where each increment in pitch is judged by listeners to be equal in distance from one another.

In Figure 1 we can see the spectrogram of a single 0.97 second segment from the GZTAN datset. This is a segment from a track classified as metal, when listened to can clearly be identified as a drum beat, then two guitar notes. These three features are visible on the spectrogram; going along the x axis, at the start there is a bright area near the bottom of the graph in the low frequency region corresponding to the lower pitch drum, then there are the two areas that start bright and fade in the higher frequency region between 2048Hz and 4096Hz.

In Figure 2 we can see the spectrogram of a different segment, this time with the words ”one love” sang from halfway through the sample. The harmonic frequencies present in the sound of the voice are visible as the multiple horizontal bright lines going up the y axis. The end of the word ”love” is cut off at the end of the sample, this can be seen as the second set of bright horizontal lines are shorter than the first set.

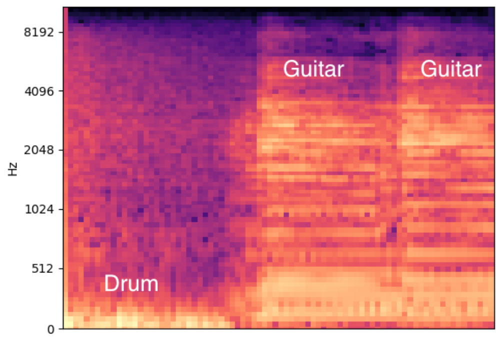


Fig. 1: Spectrogram of a drum beat then 2 guitar notes.

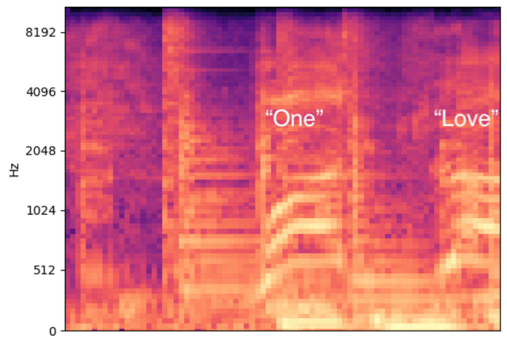


Fig. 2: Spectrogram of the words ”one love”.

V. Method (Schindler et al.)

The paper specifies the architecture of two neural networks, one shallow and one deep, that each extract features from log-mel-scaled spectrograms calculated from samples from song databases. Both architectures use two parallel pipelines that are optimised to capture either time or frequency features, building on the related work discussed earlier. The paper aims to compare the differences in accuracy between the two architectures, between different lengths of training, and between the original dataset and an augmented one.

The paper showed results using raw, maximum probability and majority vote prediction accuracy. The raw prediction is simply the highest likelihood class of each sample. Maximum probability prediction takes the sum of the likelihoods for each class across all samples from a single track then takes the highest summed likelihood. Majority vote prediction takes the raw predictions for all samples from a single track and sums the number of time each class is predicted.

On the GZTAN dataset the paper's results show there was not a significant improvement in accuracy when using deep networks over shallow ones on smaller original datasets. The paper found augmentation improved the performance of the deep neural network, however hurt the performance of the shallow one. The paper also did not find a significant improvement for the deep network when comparing data augmentation with longer training on the original dataset.

VI. Implementation Details

To replicate the architecture we used Tensorflow 1.2. This version of Tensorflow does not have a version of LeakyReLU built in, therefore we used plain ReLU as the activation function in each layer where LeakyReLU was specified by the paper. We reasoned that this shouldn't have a large effect on the results from the conclusions of the related work referenced earlier. [3]

We created the architecture as specified in the paper, using tf.layers functions conv2d, dense, and max_pooling2d. The weights initialisation is not specified in the paper. We chose to use Xavier initialisation as this ensures the random weights initially chosen are the of the right size to ensure the signal stays within a reasonable range of values as it travels through the layers. [5]

For each max pooling layer, we used a stride length equal to the size of the max pooling filter, so that we did not have any overlap between the filters.

Each epoch, the mel-spectrograms of all samples in the training dataset are presented to both pipelines of the networks. When we attempted to run this initially we ran out of memory. The test and training datasets are evenly divisible by 15, therefore we split the training data into 750 mini-batches of size 15 and iterated over every mini-batch in the training set each epoch.

Each epoch the training loss is summed across all training batches and raw accuracy on the validation set is calculated by averaging the raw accuracy across all test batches.

We created a helper class to deal with loading and batching the data, applying the melspectrogram function, and extracting the samples from each track. Upon loading the data the melspectrogram function is applied over the whole set of samples using numpy's apply_along_axis function. This sped up processing time compared to using a list comprehension.

In order to create a graph for computing the maximum probability prediction as defined in the paper, we apply the function tf.reduce_sum to the neural network graph along the 0th axis, to compute a vector of summed probabilities for each class. We can then use tf.argmax on this vector to get the label with the highest overall probability.

In order to create a graph for computing the majority vote prediction as defined in the paper, we apply the function tf.argmax to the neural network graph along the 1st axis to get the raw prediction of each sample. We then apply tf.bincount to create a vector of the number of occurrences of each predicted label. We can then use tf.argmax on this vector to get the label with the most votes.

To compute the accuracy of these predictions; we run the graph once for every track, each epoch presenting the graphs with a tensor of every sample from the respective track. The label predicted by each graph is taken and compared with the true label for the respective track. The number of matches (i.e. correct predictions) is divided by the number of tracks to give the accuracy of the two prediction methods.

In order to analyse the results in more detail, in particular the miss-classifications, we added a function to our helper class that would output original samples in .wav format. We then tested each track for various criteria, including when the maximum probability and majority vote predictions disagreed, or were both incorrect. Then we calculate the confidence values for each sample in this incorrectly classified track, and output these along with the spectrograms and .wav files.

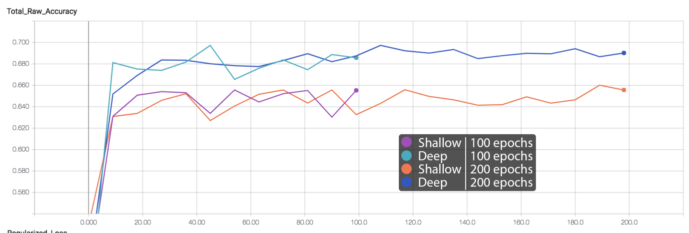
To compute the confusion matrix we created a 10x10 zeroed matrix, and incremented the relevant location in this matrix by 1 for each track's true label and maximum probability prediction.

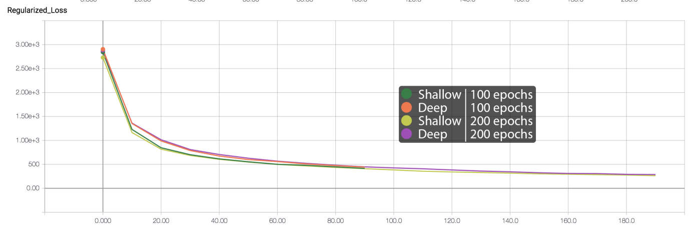
VII. Replicating Results

We found our results were all within 5% of those given.

	Model	raw	max	maj	ep
	shallow	65.20	81.20	78.90	100
	deep	68.60	83.20	82.40	100
	shallow	64.80	79.60	78.80	200
	deep	68.60	84.80	83.20	200

VIII. Training Curves





We can see here that despite the loss decreasing throughout for all models, the validation accuracies reach a plateau very quickly, with little improvement past 40 epochs. This suggests the models are overfitting the training data. From 100 to 200 epochs the shallow model decreases in accuracy and the deep model stays the same accuracy.

IX. Qualitative Results

In Figure 3 we can see an example of a spectrogram from track 58 that has been incorrectly classified by both the raw and maximum probability predictions as rock, however has been correctly classified by the majority vote prediction as metal. This is due to the majority of the samples from the

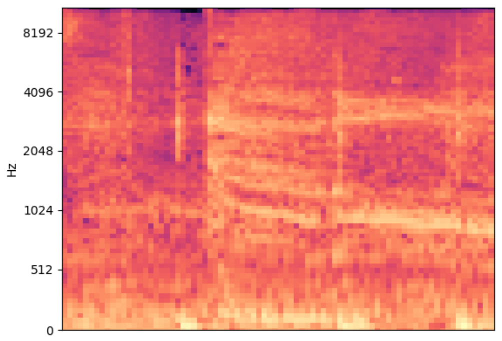


Fig. 3: Spectrogram of sample from metal track 58 incorrectly classified as rock.

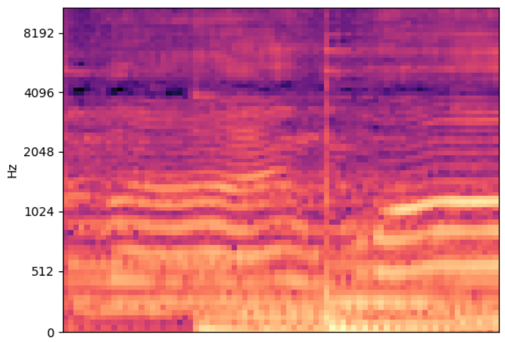


Fig. 4: Spectrogram of sample from pop track 7, where correct class has very low confidence.

track being correctly classified as metal, however the ones that have been classified incorrectly as rock have had a very high confidence, so their summed likelihoods have been greater than the summed likelihood of the metal classifications. This is not an entirely unexpected result, as metal music is a sub-genre of rock music and the two share a lot of musical similarities, in instruments used, rhythmic patterns, and tempo.

In Figure 5 we can see an example of a spectrogram from track 4 where all 3 prediction types were incorrect. The track is metal, however for this sample the highest confidence classes were country, then reggae, then metal, then rock. Overall the maximum probability and majority vote predictions for this track were both rock, which again is a reasonable prediction for the reasons listed before. The reasons for the confidence of the correct class being 3rd highest is likely due to the lack of distinguishable features in this spectrogram. The single vertical line in the middle indicates a note, however there are not any high amplitude harmonic frequencies visible. As before, the confidence in the prediction for metal is very similar to the confidence in the prediction for rock.

In Figure 4 we can see an example of a spectrogram from track 7, where the confidence in the correct class is very low. This was the output of the shallow network trained for 100 epochs. The confidence values are [-4.15, 2.48, 8.06, 2.75, -3.78, 6.11, -13.15, **-4.47**, 3.86, 0.19]. The correct class 7 (pop) has a confidence value of -4.47, which is the 9th lowest confidence out of all the classes.

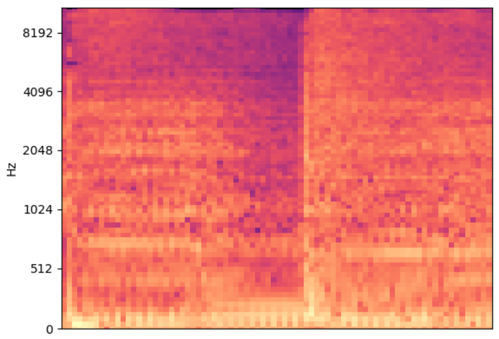


Fig. 5: Spectrogram of sample from metal track 4 with the correct class having the 3rd highest confidence.

The top class prediction was country.

X. Improvements

A. Exponential Decay Learning Rate (EXP)

In the paper the learning rate used is constant and set to $5 * 10^{-5}$. In the training curves we can see the model rapidly trains at the start then slows down to a near plateau after just 20 epochs. This likely means the learning rate is too high, therefore it is quickly moving towards the global minimum but is overshooting it as it approaches. We therefore propose changing this to an exponentially decaying learning rate. This can help the algorithm to converge earlier, as initially it moves faster towards the minima, then as we approach the minima the learning rate reduces so that we avoid overshooting.

Here we calculate the learning rate each epoch as

$$lr = lr_0 * rate^{\frac{global}{decay}}$$

Where lr_0 is the initial learning rate, $rate$ is the factor to decrease by every $decay$ number of epochs, and $global$ is the current epoch number. We started at $5 * 10^{-5}$, decreasing by a factor of 0.9 every 3000 epochs.

B. Contant-Q Transformation (CQT)

In the related paper by M. Huzaifah [2], it is posited that a CQT spectrogram representation of the audio data would perform better than a mel-spectrogram representation of the audio data due to it preserving harmonic structure even as the fundamental frequency changes. The harmonic structure is characteristic of the timbre of an instrument, which may help differentiate different instruments more than just scaling pitches perceptually as per the mel scale. The frequencies are still scaled, but according to octaves rather than listener perception.

To perform this operation we used the librosa function core.cqt, with the same hop length and number of bins as before. This gave us an 80x80 spectrogram which was fed into the neural networks. This comes with the disadvantage of being much slower to calculate than an FFT with a mel filter applied as before, this can be nullified by constructing a kernel such that it can be applied to an FFT as before. [4]

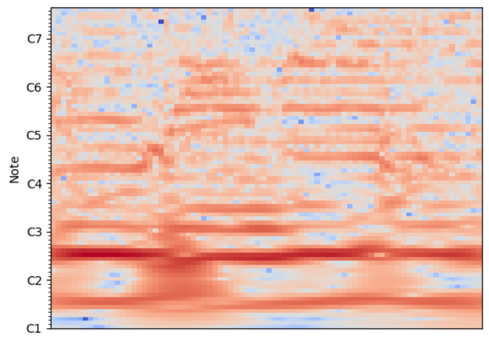


Fig. 6: CQT spectrogram of sample from track 0 scaled to octaves.

	bl	cl	co	di	hi	ja	me	po	re	ro
blues	20	0	1	2	0	0	0	1	0	1
classical	0	23	0	0	0	2	0	0	0	0
country	0	0	23	0	0	0	2	0	0	0
disco	0	0	1	23	0	0	0	0	0	1
hiphop	0	1	0	0	21	1	0	1	1	0
jazz	0	1	0	0	0	24	0	0	0	0
metal	0	0	0	1	1	0	21	0	0	2
pop	0	0	2	0	0	1	0	20	0	2
reggae	0	0	0	0	2	0	0	1	22	0
rock	0	1	2	0	0	0	2	3	0	17

Fig. 7: Confusion matrix, ground truths on the left column and maximum probability predictions along the top.

We pre-compute the spectrograms so this time cost is not an issue.

In Figure 6 a CQT spectrogram is displayed with the y axis scaled to notes, each major increment being an octave apart. The repeating horizontal bands characterise chords being played.

Through testing we found this representation gave better results with a slightly lower learning rate; $3 * 10^{-5}$ worked best.

C. Results

Our CQT extension improved the majority vote accuracy on the shallow network, over both 100 and 200 epochs, by 4.0 and 4.8 percentage points respectively. Our EXP extension produced 6.0 and 5.6 percentage point improvements. Applying both extensions produced 8.0 percentage point improvements.

Our EXP extension improved the maximum probability accuracy on the deep network over 100 epochs by 3.2 percentage points, however the majority vote accuracy remained the same. Our CQT extension did not improve the accuracy of the deep network on its own, however when we combined the two extensions we found an improvement in all accuracy types, over all epochs, with both shallow and deep networks. Our best model (deep, 100 epochs) produced raw, max, and maj improvements of 0.83, 4.4, and 3.6 percentage points respectively.

In general we found that both shallow and deep networks suffered from overfitting past 100 epochs, we did not find any improvements in training either deep or shallow networks with either extension.

Improvement	Model	raw	max	maj	ep
CQT	shallow	63.30	80.00	79.20	100
	deep	68.20	81.20	80.40	100
	shallow	63.90	82.00	80.80	200
	deep	67.90	82.00	82.00	200
EXP	shallow	67.60	84.40	81.20	100
	deep	69.70	84.40	81.60	100
	shallow	67.60	84.40	80.80	200
	deep	67.10	83.20	81.60	200
CQT & EXP	shallow	65.30	84.00	83.20	100
	deep	**70.00**	**85.60**	**85.20**	100
	shallow	65.00	82.40	83.20	200
	deep	69.20	85.20	84.40	200

To investigate the per class accuracies for our best model; the deep network with both CQT and exponential decay improvements implemented, trained on 100 epochs; we computed the confusion matrix shown in Figure 7 by comparing the maximum probability predictions with the ground truths for each track in the test dataset. From this it can be observed that our best model excels for the classes classical, country, disco and jazz, with accuracies of 92% to 96%. The largest confusions are between the classes rock, pop and blues.

XI. Conclusion and Future Work

Our report has detailed the full re-implementation and testing of both architectures and achieved results within 5% of the original, substantiating the claims made. We have then detailed two further adaptations, providing both theoretical justification and experimental evidence of the improvement gained. We have analysed our results in further depth than the original paper, showing the confusion matrix for the best model.

Building on this we propose more improvements could be made in restructuring aspects of the architecture. Currently the CQT adaptation uses spectrograms with 80 frequency bins, covering 7 octaves scaled by semi-tones. There are 12 semi-tones in an octave, however we are using fewer than 12 bins to represent these, so by pigeonhole principle we are losing information about exact semi-tones ($\frac{80}{7} \approx 11.4$). Therefore we propose using spectrograms with a multiple of 12 frequency bins, 84 would be the closest in this case. This may allow for more accurate feature extraction.

References

[1] Thomas Lidy, Alexander Schindler, CQT-based convolutional neural networks for audio scene classification, https://pdfs.semanticscholar.org/d468/d0a72ec686df885570706ebcd4fde261a7bb.pdf

[2] M. Huzaifah, Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks, https://arxiv.org/pdf/1706.07156.pdf

[3] Andrew L. Maas, Awni Y. Hannun, Andrew Y. Ng, Rectifier Nonlinearities Improve Neural Network Acoustic Models, https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf

[4] Judith C. Brown, An efficient algorithm for the calculation of a constant Q transform, http://academics.wellesley.edu/Physics/brown/pubs/effalgV92P2698-P2701.pdf

[5] Xavier Glorot, Understanding the difficulty of training deep feedforward neural networks, http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdfn