COMP47650 Deep Learning Project

Niamh Belton
niamh.belton@ucdconnect.ie

Abstract

This work investigates Deep Learning techniques for Music Genre Classification (MGC). Four experiments were conducted on the open-source GTZan dataset. The first experiment involved training a Multi-Layer Perceptron on tabular data. The remaining experiments focused on representing the audio signal in the form of Mel Spectrograms. These Mel Spectrograms were then input into Convolutional Neural Networks (CNN). This analysis found that the ResNet18 architecture achieved the highest accuracy for MGC and the model performance is further increased with the addition of domain specific data augmentation techniques.

1 Introduction

Music Genre Classification (MGC) is an area in the field of Music Information Retrieval (MIR). This is a field that analyses audio for various tasks such as music similarity and artist identification (Grzywczak and Gwardys, 2014). MGC is concerned with automating the classification of audio into music genres. This can be done by extracting features from the audio and employing classification models. In this work, I investigate various methods of automating MGC with Deep Learning methods. This paper is organised as follows; Section two is related work, Section three is the materials used, Section four is experimental set up, Section five is the results and Section six is the conclusion and future work.

2 Related Work

The GTZan dataset is a well known benchmark dataset for Music Genre Classification (MGC) since it was originally released in 2001 (Tzanetakis and Essl, 2001). Initial approaches focused on extracting features from the audio signal such as 'Zero-Crossing Rate' in the time domain and 'Spectral Centroid' and 'Spectral Rolloff' in the frequency domain (Gwardys and Grzywczak, 2014), (Grzywczak and Gwardys, 2014). Once these hand crafted features were engineered, many studies applied Machine Learning techniques such as boosting (Bergstra et al., 2006) and Support Vector Machines (SVM) (Li et al., 2003) to automate MGC. Many of these studies achieved accuracies greater than 75% (Panagakis et al., 2008).

More complex features can be extracted from the audio signal. These include Spectrograms, Chromograms, Mel-Frequency Cepstral Coefficients (MFCC) and Mel Spectrograms (Gwardys and Grzywczak, 2014). These representations are 2D and therefore, they can be interpreted as images. Hence, this makes the use of Convolutional Neural Networks (CNN) possible for audio classification tasks. Choi et al. (2017) used Mel Spectrograms and a CNN that achieved 89.9% accuracy. More recently, Bian et al. (2019) achieved state-of-the-art performance by converting the audio signal to spectrograms and employing a DenseNet and ResNet.

Other common approaches in the literature for improving audio classification include transfer learning and data augmentation. Choi et al. (2017) used a pre-trained model on audio data, while Gwardys and Grzywczak (2014) used pre-trained model on the ILSVRC-2012 dataset. This technique improves training. Moreover, Bian et al. (2019) and Aguiar et al. (2018) proved the effectiveness of including domain specific augmentation techniques such as adding noise to the audio signal. Data augmentation increase the size of the dataset and therefore, it aids the development of robust models.

3 Materials

The GTZAN dataset was used for this analysis (Olteanu et al., 2020). The dataset, originally proposed by Tzanetakis and Essl (2001), consists of 1,000, 30 second audio clips. Each clip is labelled as belonging to one of ten possible genres. This a balanced dataset and each audio clip has a sampling rate of 22,050 Hz. The dataset includes the raw audio data in .wav format, images of the Mel Spectrograms and tabular data that includes various hand engineered features. There are no details provided on how the Mel Spectrogram images were generated.

4 Experimental Set Up

The following section outlines four experiments conducted as part of this analysis. The first experiment focuses on the tabular data and the remaining experiments focus on converting the audio to images and using CNNs. The Pytorch library (Paszke et al., 2019) was used to implement the models.

4.1 Train, Validation and Test Split

The dataset was divided into train, validation and test sets in the radio 3:1:1. The data was shuffled with a random seed equal to zero and the last 20% of the data was used as the test set. Another 20% was used as the validation set. This resulted in slightly uneven class balances across the splits. This can be seed in Figure 1. All models were trained on the training data. The validation data was then used to select the best model and the best model was assessed on the test data.

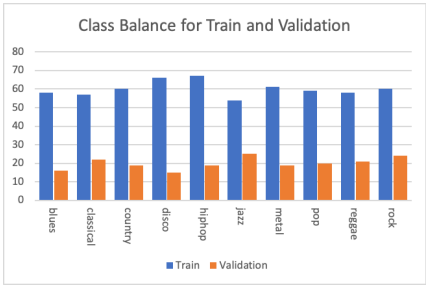


Figure 1: Class balances of train and validation set.

4.2 Experiment 1 - Multi-Layer Perceptron

The open-source dataset includes tabular data. This data consists of 57 hand crafted features that provide information about each audio signal. A Multi-Layer Perceptron (MLP) was implemented to classify the music genre. MLPs have an input layer, hidden layers with varying numbers of neurons, activation functions and an output layer. Figure 2 outlines the different architectures and hyper-parameters assessed for this task. The weights were initialised with values drawn from uniform distribution where the mean is zero and the standard deviation is equal to $m^{1/2}$ where m is the number of connections feeding into the neuron (LeCun et al., 1998). This is the default method in Pytorch. The data was standardised by subtracting the mean value for each feature in the training data and dividing by the standard deviation. The number of epochs was set to a maximum of three hundred and the epoch with the highest validation accuracy was recorded. The Negative Log Likelihood (NLL) Loss function was used for training.

Number of Hidden Layers	Number of Neurons	Activation Function	Batch Size	Learning Rate	Epochs	Accuracy on Validation Data
1	32	ReLU	32	0.001	221	47.0%
1	100	ReLU	32	0.001	111	46.0%
2	1st Layer - 128 2nd Layer - 64	ReLU	32	0.001	120	46.5%
1	32	ReLU	8	0.001	227	47.0%
1	100	ReLU	8	0.001	53	49.5%
2	1st Layer - 128 2nd Layer - 64	ReLU	8	0.001	112	46.5%
1	32	ReLU	8	0.0001	143	46.0%

Figure 2: Architecture and hyper-parameters for MLP.

Although the accuracy is poor on the validation data, it is evident that the model is training and learning relationships in the data as the accuracy is significantly higher than randomly guessing.

Randomly guess would yield an accuracy approximately equal to 10%, the class balance. Given that the model was incorrectly classifying over half of the cases, no further tuning or investigation was conducted on this model.

4.3 Experiments 2-4 - CNNs and Images

4.3.1 Architecture

Experiments 2-4 focus on converting the audio to images and employing a CNN as this is the current state-of-the-art technique for audio classification and MGC. In this analysis, I investigated two classic architectures; AlexNet (Krizhevsky et al., 2012) and ResNet18 (He et al., 2015). The AlexNet architecture was one of the first deep neural networks. At the time it was initially proposed, it won the ILSVRC-2012 competition by a significant margin (Russakovsky et al., 2015). The architecture consists of four main blocks where each block consists of 2D convolution, a ReLU activation function and a max pooling layer. This is followed by a Global Average Pooling (GAP) layer and two fully connected layers with ReLU activations. In the final two fully connected layers, it uses a technique known as Dropout. This technique randomly drop neurons during training. In this architecture, each neuron has a probability of 0.5 of being dropped. This acts as a form of regularization and therefore, it prevents over-fitting (Srivastava et al., 2014).

It was known that while deeper architectures have the ability to perform better and discover more complex features, they suffered from vanishing or exploding gradients during back-propagation (Glorot and Bengio, 2010). The ResNet overcame this limitation by using a skip connection. A skip connection feeds the output of a layer into the next layer and a layer further on in the network. The ResNet also uses a technique known as batch normalisation in every layer (Ioffe and Szegedy, 2015). This normalises the batch at different stages in the network which enables faster convergence and further reduces the challenge of vanishing or exploding gradients. The ResNet comes in many different forms depending the number of layers; ResNet18, ResNet34, ResNet101 and many more. This work focuses on the ResNet18 which consists of four main layers, where each layer is a residual block. This is followed by a GAP layer and fully connected layers.

Larger Deep Learning models were excluded from the analysis as the size of the dataset in question is small and therefore, larger deep learning networks would easily over-fit to the data. I experimented with the deep VGG16 network (Simonyan and Zisserman, 2015) on the GTZan dataset. Despite being a significantly larger network, it resulted in a validation accuracy of 81.5% which is a similar performance to the AlexNet and ResNet18 architectures that achieved 80% and 86% respectively. (These results are based on the methodology outlined in Section 4.3.4). Therefore, no further analysis was conducted on this architecture or deeper networks. Table 1 shows each of the architectures and the number of trainable parameters. Trainable parameters are the parameters that are affected by backpropagation.

Table 1: Architecture Comparison (Anwar, 2019)

Model	**Number of Trainable Parameters (Million)**
AlexNet	$\approx 62m$
ResNet18	$\approx 11.5m$
VGG16	$\approx 138.5m$

I opted to implement classic architectures rather than developing a new one so that I could make use of pre-trained models. One of the primary challenges with this dataset is its small size. Many of the classic architectures have widely available pre-trained weights on the ImageNet dataset. By initialising the model with pre-trained weights, it transfers low-level features such as edges to the task in question. This enables faster convergence and better performance. An experiment was conducted to assess if the network could be trained from scratch. The weights were initialised using the method outlined in Section 4.2 (LeCun et al., 1998). Using the a ResNet18 architecture and the method outlined in Section 4.3.4, it achieved a poor accuracy on validation data of 53.5%. This justifies the use of pre-trained models for this task.

4.3.2 Implementation Details

- The weights were initialised with the pretrained model's weights. All weights were then subsequently fine-tuned.
- The Adam optimiser was employed to update network weights (Kingma and Ba, 2017). Adam was chosen as it is known to be fast, achieve good results and outperform other common optimisers such as RMSProp (Ruder, 2016).

- The Negative Log Likelihood (NLL) loss was chosen as the loss function as this is a common loss function for multi-class classification. The NLL loss takes values that were output from a log Softmax function as input (*NLLLOSS*, n.d.). Therefore, a log Softmax layer was added to the model. To calculate the accuracy, the class with the largest output from the log Softmax layer is considered to be the model's predicted class.
- Default Hyper-parameters
  - The learning rate determines the step-size of the optimisation algorithm at each iteration. The default value for all experiments was 0.00001. However, this value varied with experiments.
  - The batch size is the amount of data that was used in one iteration of the algorithm. This was set to 32 as the dataset is small and therefore, the batch size was also kept small. Moreover, 32 is a common batch size that is used.
  - The number of epochs is the amount of times the entire training set was passed through the model. Early stopping was used to determine the number of epochs for each experiment. This involved stopping the algorithm if there was no improvement in the validation accuracy within eight epochs. Therefore, the number of epochs was different for each experiment.
- A scheduler was used to decrease the learning rate if there was no decrease in validation loss within three epochs.

4.3.3 Experiment 2 - Provided Images

The curators of the dataset converted the raw audio data to Mel Spectrograms and saved the resulting images in .png format. No additional information was provided on how the images were generated. The images have dimensions (288, 432, 4). As the images have four channels, they use the BGRA colour model. This was converted to the BGR colour model using the OpenCV library in Python (Bradski, 2000). Thus, this converted the dimensions to (3, 288, 432). The pixel values ranged from zero to 255. The pixels were normalised by dividing them by 255. Although the pre-trained AlexNet expects (3, 227, 227) and ResNet18 expects (3, 256, 256) input dimensions, the GAP layer allows for flexibility in the input dimensions, meaning the images did not have to resized. Therefore, the only modifications to the network were editing the last fully connected layer to output a one-dimensional vector of size ten, a value for each class and adding a log Softmax layer.

This method uses all default parameters as outlined in section 4.3.2. The number of epochs that resulted in the best validation accuracy was 17 for AlexNet and 16 for ResNet18.

4.3.4 Experiment 3 - Curating Images

The details of how the images in the dataset were generated are not outlined. Therefore, I converted the audio signal to Mel-Spectrograms. These can then be used as images in 2D CNNs. The raw audio signal in the dataset is given in the time domain. An example of this is visualised in Figure 3(a). The y-axis shows the amplitude of the audio signal and the x axis shows the time. This signal is made up of 'single-frequency sound waves' (Chaudhary, 2020). I used the Short-Term Fourier Transform (STFT) to decompose the audio signal into its frequency components. This converts the data from the time domain to the frequency domain. I used a window length of 1,024 and a hop length of 512. This means the STFT is calculated for a segment of data of length 1024. The window then slides across 512 elements in the data and the STFT is re-calculated. The data is then visualised in the frequency domain in Figure 3(b). They y-axis shows the magnitude of each frequency.

It is known that humans can easily decipher between frequencies at lower frequencies but it is more difficult to differentiate between higher frequencies. For example, a human can easily hear the difference in 100Hz and 200Hz but they would struggle to decipher between 1000 Hz and 1100 Hz (Sabra, 2021). The Mel scale transforms the frequency so that the distance between frequencies is more similar to how humans interpret the distance between frequencies.

Although interpreting the audio signal in the frequency domain with Mel Scaling provides insight into the audio signal, it is important to preserve the sequential element of the audio signal and to understand the order that the audio occurred. Mel Spectrograms are a visualisation technique that shows the data in both the frequency and time domain. It also converts the amplitude of the frequency to decibels. Therefore a Mel spectrogram shows the time on the x axis, the Mel Scale on the y axis and the values are colour coded depending on the decibel value. The resulting Mel Spectrograms can

then be used as images in CNNs. Figure 3(c) shows an example of Mel Spectrogram generated using the 'librosa.specshow' function in the librosa library.

The Mel Spectrograms had dimensions (256, x) where x varied from 1292 to 1320. Images were padded with zeros so that all images had dimensions (256, 1320). The images were then stacked on top of each other to obtain dimensions of (3, 256, 1230). After the images were converted to Mel Spectrograms, they were standardised by taking away the minimum pixel value in the training data and dividing by the maximum pixel value in the training data. Therefore, all pixels had a value between zero and one.

This method uses all default parameters as outlined in section 4.3.2. The number of epochs that resulted in the best validation accuracy was 19 for AlexNet and 12 for ResNet18.

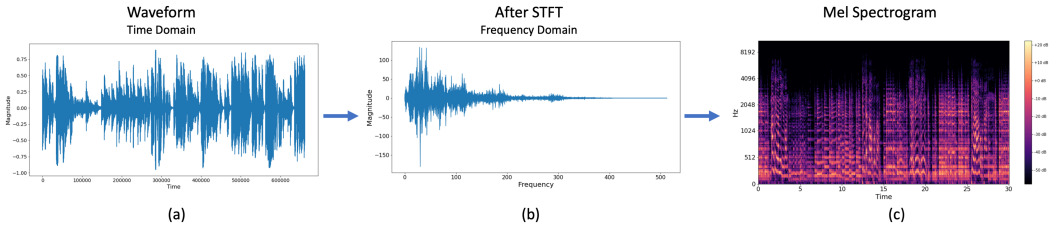


Figure 3: (a) Raw audio signal in time domain. (b) Magnitude of frequencies in frequency domain. (c) Mel Spectrogram.

4.3.5 Experiment 4 - Data Augmentation

Given that the ResNet18 performed the best in experiment three, experiment four involves implementing the ResNet18 with data augmentation. Implementing data augmentation has two main advantages. The first is that it increases the size of the dataset. This is particularly useful in this task as the data set is small. Secondly, adding noise to the dataset creates a more robust model and it will preventing over-fitting. The typical affine data augmentation techniques such as flipping or rotating are not suitable in this context. Therefore, domain specific augmentation techniques were employed. Aguiar et al. (2018) outlined various effective data augmentation techniques for audio data. The augmentation techniques that were implemented are outlined in the bullet points below.

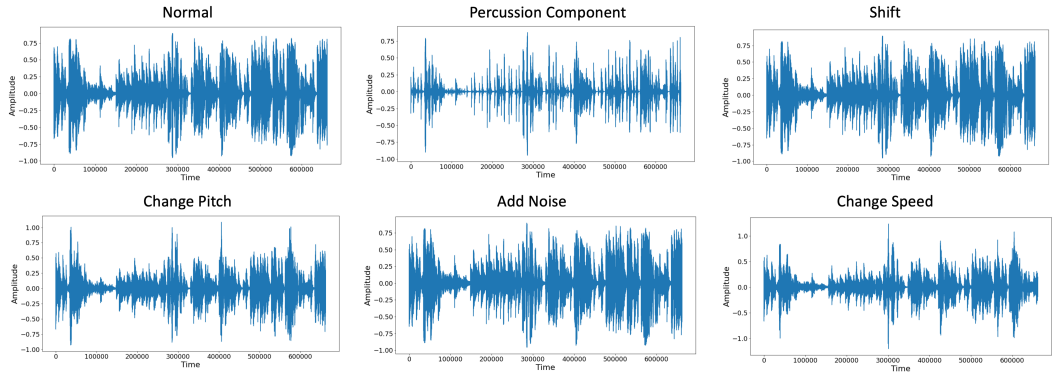


Figure 4: Different Augmentation Techniques on an example case.

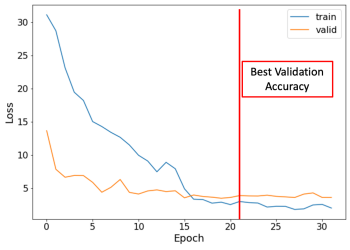
- Percussion Component - Extracts only the percussion component of the audio.
- Adding noise - Random data was generated from the standard normal distribution. The number of random data points generated was equivalent to the length of the audio wave. In order to not majorly disrupt the audio signal, 0.005 of the generated noise was added to the original sound wave.
- Shift - The raw audio signal was shifted 1,600 places to the left. The first 1,600 values were then equal to the last 1,600 places.
- Change Pitch - Changes the pitch of audio by a random amount of steps.
- Change Speed - Change the speed by a random value between 0.9 and 1.1.

Figure 4 shows an example of an audio wave belonging to the class 'blues' and how the different augmentations affect it. All of these techniques were applied to the raw audio data before transforming it to Mel Spectrogram images. All data augmentation had a probability of 0.25, meaning that multiple augmentation techniques could be applied to the audio. However, the percussion augmentation was always applied last as further augmentation after extracting the percussion component could majorly disrupt the data.

Figure 5(a) shows the output dimensions for each layer of ResNet18 architecture. Figure 5(b) shows the total loss for each epoch for the train and validation data. As expected, the training loss steadily decreases to zero, while the validation loss plateaus earlier. The best validation accuracy is at the 22nd epoch. The model is overfitting to the training data for the remaining epochs. Figure 5(c) shows the hyper-parameters that were assessed. A batch size of 32 and a learning rate of 0.001 performed the best on the training data.

Layer	Output Dimensions
Input	(3, 256, 1320)
Conv2D, Batch Normalisation, Relu,	(64, 128, 660)
Maxpool	(64, 128, 330)
Residual Block 1	(64, 64, 330)
Residual Block 2	(128, 32, 165)
Residual Block 3	(256, 16, 83)
Residual Block 4	(512, 8, 42)
Global Average Pooling	(512)
Fully Connected Layer	(10)

(a)



(b)

Batch Size	Learning Rate	Epochs	Validation Accuracy
32	0.0001	22	87.5%
32	0.00001	6	75%
32	0.001	1	36%
8	0.0001	23	85%

(c)

Figure 5: (a) Output Dimensions of Each Layer. (b) Train and Validation Loss. (c) Hyper-parameters.

5 Results

The models were evaluated quantitatively on the validation data in order to select the best model to test on the test set. Table 5 shows the accuracy for each experiment on the train and validation set. As experiment four (outlined in Section 4.3.5) performed marginally better than other methods, this was selected as our final model. It achieved 77.39% on the test set. It is expected that experiment four would outperform other methods for two main reasons. The first of which is that the ResNet18 architecture is known to outperform AlexNet in many image classification tasks and secondly, the addition of data augmentation increases the robustness of the model.

Table 2: Model Performance Comparison

Experiment	**Train Accuracy**	**Valid Accuracy**	**Test Accuracy**	**Train Time**
Exp1 MLP	95.64%	46.5%		<2
Exp2 AlexNet	91.85%	74.87%		3
Exp2 ResNet18	100%	74.87%		3
Exp3 AlexNet	97.83%	80%		42
Exp3 ResNet18	99.67%	86%		42
Exp4 ResNet18	94.67%	**87.5%**	**79.29%**	310

Table 5 also reports the training time on a Tesla GPU in minutes. The number of epochs was different for different experiments due to the early stopping. The data augmentation significantly increased the training time. Although experiment four performed the best on validation data, it is more computationally expensive. Therefore, experiment three with ResNet18 architecture which achieved a similar accuracy to experiment four may be the optimal choice in some cases.

6 Conclusion and Future Work

In this work, we have implemented pre-trained classic Deep Learning architectures. This analysis found that a ResNet18 architecture with domain specific data augmentation techniques achieve good performance on the dataset. However, the addition of data augmentation increases the training time seven fold and therefore, the trade-off between model performance and computational time should be considered. One of the primary challenges of this data was its small size. Therefore, future work will include splitting up the audio clips into smaller clips in order to increase the size of the dataset. With a bigger dataset, I can implement more techniques such as a Recurrent Neural Network that will take into account the sequential nature of the audio signal. Moreover, it was found that the model outlined

in experiment four could classify each class at rates greater than 76% with the exception of the 'rock' genre. This had a classification rate of 43.75%. Future work will focus on why the model performs poorly for this class.

References

Aguiar, L. R., Costa, M. G. Y. and Silla, N. C. (2018), 'Exploring data augmentation to improve music genre classification with convnets', *2018 International Joint Conference on Neural Networks (IJCNN)* pp. 1–8.

Anwar, A. (2019), 'Difference between alexnet, vggnet, resnet, and inception', `https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96`.

Bergstra, J., Casagrande, N., Erhan, D., Eck, D. and Kégl, B. (2006), 'Aggregate features and a da b oost for music classification', *Machine learning* **65**(2-3), 473–484.

Bian, W., Wang, J., Zhuang, B., Yang, J., Wang, S. and Xiao, J. (2019), 'Audio-based music classification with densenet and data augmentation'.

Bradski, G. (2000), 'The OpenCV Library', *Dr. Dobb's Journal of Software Tools* .

Chaudhary, K. (2020), 'Understanding audio data, fourier transform, fft and spectrogram features for a speech recognition system'.
URL: *https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520*

Choi, K., Fazekas, G., Sandler, M. and Cho, K. (2017), 'Transfer learning for music classification and regression tasks'.

Glorot, X. and Bengio, Y. (2010), Understanding the difficulty of training deep feedforward neural networks, *in* Y. W. Teh and M. Titterington, eds, 'Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics', Vol. 9 of *Proceedings of Machine Learning Research*, PMLR, Chia Laguna Resort, Sardinia, Italy, pp. 249–256.
URL: *http://proceedings.mlr.press/v9/glorot10a.html*

Grzywczak, D. and Gwardys, G. (2014), Audio features in music information retrieval, *in* D. Ślzak, G. Schaefer, S. T. Vuong and Y.-S. Kim, eds, 'Active Media Technology', Springer International Publishing, Cham, pp. 187–199.

Gwardys, G. and Grzywczak, D. (2014), 'Deep image features in music information retrieval', **vol. 60**(No 4).
URL: *http://journals.pan.pl/Content/87690/PDF/eletel-2014-0042.pdf*

He, K., Zhang, X., Ren, S. and Sun, J. (2015), 'Deep residual learning for image recognition', *CoRR* **abs/1512.03385**.
URL: *http://arxiv.org/abs/1512.03385*

Ioffe, S. and Szegedy, C. (2015), Batch normalization: Accelerating deep network training by reducing internal covariate shift, *in* 'Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37', ICML'15, JMLR.org, p. 448–456.

Kingma, D. P. and Ba, J. (2017), 'Adam: A method for stochastic optimization'.

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, *in* F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger, eds, 'Advances in Neural Information Processing Systems', Vol. 25, Curran Associates, Inc.
URL: *https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf*

LeCun, Y., Bottou, L., Orr, G. B. and Müller, K.-R. (1998), Efficient backprop, *in* 'Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop', Springer-Verlag, Berlin, Heidelberg, p. 9–50.

Li, T., Ogihara, M. and Li, Q. (2003), A comparative study on content-based music genre classification, SIGIR '03, Association for Computing Machinery, New York, NY, USA, p. 282–289.
URL: *https://doi.org/10.1145/860435.860487*

NLLLOSS (n.d.), `https://pytorch.org/docs/master/generated/torch.nn.NLLLoss.html#torch.nn.NLLLoss`.

Olteanu, A., Wiltshire, J., O'Hare, L. and Lei, M. (2020), 'Gtzan dataset - music genre classification'.

Panagakis, I., Benetos, E. and Kotropoulos, C. (2008), Music genre classification: a multilinear approach, *in* J. Bello, E. Chew and D. Turnbull, eds, 'International Symposium Music Information Retrieval', pp. 583 – 588.
URL: *https://openaccess.city.ac.uk/id/eprint/2109/*

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. and Chintala, S. (2019), Pytorch: An imperative style, high-performance deep learning library, *in* H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, eds, 'Advances in Neural Information Processing Systems 32', Curran Associates, Inc., pp. 8024–8035.
URL: *http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf*

Ruder, S. (2016), 'An overview of gradient descent optimization algorithms', *CoRR* **abs/1609.04747**.
URL: *http://arxiv.org/abs/1609.04747*

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. and Fei-Fei, L. (2015), 'ImageNet Large Scale Visual Recognition Challenge', *International Journal of Computer Vision (IJCV)* **115**(3), 211–252.

Sabra, A. (2021), 'Learning from audio: The mel scale, mel spectrograms, and mel frequency cepstral coefficients'.
URL: *https://towardsdatascience.com/learning-from-audio-the-mel-scale-mel-spectrograms-and-mel-frequency-cepstral-coefficients-f5752b6324a8*

Simonyan, K. and Zisserman, A. (2015), Very deep convolutional networks for large-scale image recognition, *in* 'International Conference on Learning Representations'.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014), 'Dropout: A simple way to prevent neural networks from overfitting', *J. Mach. Learn. Res.* **15**(1), 1929–1958.

Tzanetakis, G. and Essl, G. (2001), Automatic musical genre classification of audio signals, *in* 'IEEE Transactions on Speech and Audio Processing', pp. 293–302.