SoundFusion: Music Recognition by Information Fusion of Different Audio Representations

George P. Prodan †, Melika Keshavarz†, Lorenzo Ausilio†

Abstract—**We present a comprehensive study of the use of Convolutional Neural Networks (CNNs) in music classification, a rapidly growing field in computer science. Our aim is to provide an in-depth analysis of the various approaches and architectures that have been developed for music classification. We propose SoundFusion, a model that implements information fusion by using two different residual networks. We utilize both raw audio signals and spectrogram representations as inputs, with the goal of accurately classifying music into eight predefined genres. We analyze possible ways to pre-process the data before feeding it to the model, and a comparative analysis of the state-of-the-art models on the FMA small dataset. We show that regularization techniques and width scaling were effective in improving the performance of CNNs in a limited data scenario and that the proposed SoundFusion architecture outperformed the baseline models.**

I. INTRODUCTION

Music information retrieval (MIR) is a rapidly growing field in computer science that aims to develop computational methods for automatically analyzing and organizing large collections of music. Music genre classification, as a subfield of MIR, has recently gained traction. The vast amounts of music available online and the growing number of music streaming services have led to the need for more sophisticated methods for classifying music. Convolutional Neural Networks (CNNs) have recently emerged as a state-of-the-art method for solving various computer vision problems and are gaining traction in the field of music classification as well.

In this paper, we delve into the exciting world of music classification - a challenge that requires the identification of a musical genre or style. We aim to review the current state-of-the-art in using CNNs for music classification and provide an in-depth analysis of the various approaches and architectures that have been proposed for this task. We approach this task in a dual way, by utilizing the information provided by the raw signal itself and also by taking advantage of the spectrogram representation of the tracks, leveraging the powerful capabilities of Convolutional Neural Networks (CNNs) as image classifiers.

We present a comprehensive music classification system using various CNN architectures implemented in PyTorch, including ResNet34 and ResNet18 (both with pretrained and random weight initialization) [1] [2], nnet1, nnet2 [3], and our own custom ResNet34 architecture with added regularization techniques for improved performance. Additionally, we propose our own Simple CNN for comparison.

Our approach utilizes both raw audio signals and spectrogram representations as inputs to the CNNs, with the goal of accurately classifying music into eight predefined genres.

The focus of our paper is twofold: Firstly, the creation of our own custom preprocessing class to prepare the data before feeding it to the model, and secondly a comparative analysis of the state-of-the-art models on the FMA small dataset. Performance of our models are analyzed through observation of correct music genre predictions as well as confusion matrices to showcase the effectiveness and shortcomings of using CNNs in music classification tasks.

Our results offer valuable insights into the potential of deep learning techniques in music classification and pave the way for exciting future research opportunities in this field.

†University of Padova, Department of Physics and Astronomy
email: {lorenzo.ausilio}@studenti.unipd.it

II. RELATED WORK

Music genre recognition with deep learning has been actively explored since 2010, when Li et al. have proposed a CNN model for musical pattern extraction by using Mel-frequency cepstral coefficients (MFCC) [4]. However, their model performs poorly on unseen data. Sigtia and Dixon use DNNs to extract the features from Fast Fourier transforms (FFTs) [5], their gridsearch show that ReLU + SGD optimizer lead to better performances. Another approaches of feeding the data is to use Mel Spectrograms [6] or STFT [3]. Moreover, models that process the raw audio have been developed [7]. First proposal of such a model suggests that using raw audio inputs does not lead to better performances than spectrograms, however, in this way one can discover frequency decompositions, phase- and translation-invariant features [8]. In a recent work from 2018, Pons et al [9] use a private dataset of 1.2 million songs to show that waveform-based models outperfom the spectrogram-based models at a large scale.

A. *Residual Network*

The field of Convolutional Neural Networks (CNNs) is vast and diverse, but one of its most captivating innovations is the ResNet architecture introduced by Zhang et [1] al. This revolutionary design marked a major milestone in the realm of deep architecture for image processing tasks by effectively addressing the persistent issue of "gradient vanishing" that plagued very deep, plain networks with numerous stacked layers, leading to high training error.However, in this paper the main focus is the Image processing and other types of problems, like Music Classification, are not addressed. Furthermore, Xu et al [3] proposed two groundbreaking architectures - one conventional and one residual. This study suggests that the integration of residual shortcuts and the blend of max and average pooling techniques results in an improved performance and a substantial boost in accuracy on GTZAN dataset. This study was not tested on other datasets; therefore, we do not know how these two architectures would perform for complicated and large datasets.

B. Transfer Learning

In recent years, transfer learning has become a popular approach for improving performance in audio classification tasks, especially with limited data.

Several studies have demonstrated the effectiveness of transfer learning for audio classification using CNNs. In [10], the authors proposed a transfer learning approach for music genre classification, where a pre-trained CNN was fine-tuned on a smaller dataset. The results showed that this approach outperformed training from scratch and achieved better performance on the music genre classification task. Another study [11] used transfer learning for bird species identification from audio recordings. The authors fine-tuned a pre-trained CNN on a smaller dataset and showed that this approach achieved comparable performance to training from scratch on a much larger dataset. This demonstrates the effectiveness of transfer learning for improving performance in audio classification tasks even with limited data. As result, transfer learning with CNNs has been shown to be an effective approach for improving performance in audio classification tasks, especially when limited data is available. The results from [10] and [11] suggest that transfer learning can effectively leverage knowledge learned from one task to another related task, reducing the amount of data and computational resources needed to achieve good performance in audio classification.

C. Information fusion

Information fusion in multimodal deep learning refers to the combination of information from multiple modalities, such as text, audio, image, or video, to make a more informed decision. There are different approaches such as early fusion, when the inputs are joined before feeding them to the network, or late fusion [12]. In the context of music recognition, Li et al. [13] propose a late fusion model which combines mel-spectrograms and lyric information. In this project we apply information fusion by combining waveforms and mel-spectrograms.

III. Processing Pipeline

The data processing pipeline comprises the following general steps:

1) **Data Pre-processing:** The dataset samples are pre-processed, so that we can save locally the waveforms generated from raw audio samples and spectrograms. Segmentation results are saved only for the spectrograms, because of limited storing spaces on our machines.
2) **Data Loading:** Data is loaded from the local storage and fed to the network. If it is the case, there is a probability that data is augmented.
3) **Model Training:** We run the model for 20 - 150 epochs. After each epoch, we monitorize the validation and training errors for possible overfitting or underfitting. In the case of overfitting, we use regularization techniques such as dropout layers or we increase the data augmentation probability, to make the model generalize better.
4) **Model Testing and Aggregation** We test the model on 800 samples. We show the results with and without aggregation. When using aggregation, we take into account all the results provided by different subsamples generated from the same clip. Therefore, the final output is decided by the majority class.

The data loading step may differ from one architecture to another. For the baseline architectures we use only one input. It can be a waveform or a spectrogram, and they can correspond to the full clip or to a segment of 5 seconds. The architecture developed by us takes two inputs, one waveform and spectrograms corresponding to the same segment/clip.

IV. Signals and Features

A. Raw audio representation

We take into account the following aspects before feeding raw audio inputs to the model: the number of channels, length and sample rate.

- **Channels:** An audio channel refers to a single track of audio. The final audio signal is the result of combining multiple audio channel. If the dataset comprises both mono (one channel) and stereo (two channels) signals, we convert all the samples to mono for consistency.
- **Padding or cutting:** We pad or cut the audio signal if it does not match the standard length. We compute the standard length as the product between the sample rate and the standard duration of the clips, i.e. 30 seconds. This is needed as the length of the samples is not always the same. This part of pre-processing is omitted when we use the segmentation strategy.
- **Downsampling:** We downsample the audio signals to a lower sample rate to reduce the data size and, therefore, the computational time. Downsampling is useful for tasks such as audio classification, where the lower frequencies of the signal are more important than the higher frequencies. All the samples are downsampled to a rate of 22050 Hz, the half of the original rate.

We adopt a segmentation strategy in order to increase the amount of data. We cut each clip in 5 seconds subsamples with a 25% overlap. We have not chosen a larger overlap percentage because of the lack of computational resources. As a remark, previous studies use 50% overlap [3] or 75% overlap [7], which may lead to slightly better results, but this assumption should be experimented. Another reason for using segmentation is that we can apply aggregation in predicting the final output for a clip as it will be explained later on.

B. Mel Spectrogram representation

We choose Mel spectrograms as input to our CNN as this representation has numerous advantages compared to regular spectrograms. Firstly, since we are working with limited computational resources, complexity is indeed a concern and Mel spectrograms provide a more compact representation while still maintaining a relatively high accuracy, as has been shown in certain subfields of MIR [14]. Mel spectrograms reduce the dimensionality by mapping a large number of frequencies to a smaller number of frequencies known as Mel frequencies, which are proportional to the log of the equivalent frequency in Hertz. Secondly, the Mel scale was introduced to mimic the way that humans perceive sound. Equal distances on the mel scale correspond to equal perceived changes in pitch by the human ear. As such, the mel scale compresses higher frequencies to better match human perception of sound. Thirdly, Mel spectrograms and variations of it have been widely adopted in the field of music genre classification [15], showing promising results and justifying their use case.

Similarly to the raw audio pre-processing, the generation of Mel spectrograms takes into account the following aspects : the number of channels, length, sampling rate and certain parameters defining the Mel spectrogram such as the number of mels, the length of the FFT window and the hop length.

- **Channels:** For the spectrograms we made the choice of using two channels, which means for one audio signal in the end we are going to have two spectrograms, one for each channel. This choice was motivated by the fact that computational complexity when using spectrograms is lower compared to the raw audio signal, allowing to use both channels. Thus we convert the mono samples to stereo by duplicating the already existing mono signal. Note that most tracks in the FMA dataset are encoded in stereo, thus we will rarely end up with the same spectrogram twice for one data sample.
- **Sampling rate:** As a next step we standardize the sampling rate to 44100Hz for all tracks. Reasoning behind this being that, as stated in the FMA dataset paper [16], most tracks are sampled at this sampling rate and now the constraint of computational complexity is no longer a limiting factor.
- **Padding or cutting:** When we compute the spectrograms we use the entire audio track of 30s. After having fixed the sampling rate for each track, we now also fix the length that each track should have. As most tracks are 30s in length we impose this for every track. As a consequence we either truncate the raw audio signal or pad it with zeros. We use a random padding method, meaning that if padding is applied, the sequence of silence is injected into the raw data at random points.
- **Mel spectrogram:** We generate the mel spectrograms using the built-in functions from torchaudio. We chose the number of mel filter banks to be 64, the number of samples in the FFT window is 1024 and the hop length is 512. Although the choices for the window size and hop length are many and one could experiment tuning these hyperparameters, it has been shown that regardless of the window type, an overlap ratio of 50% is adequate for music genre classification and that overlap ratios greater than 50% have no positive impact on accuracy [17]. The window function we use is the default Hann window function from torchaudio.
  Lastly we convert the amplitude scale to decibel scale to conclude the generation of our Mel spectrograms.

C. Dataset

In this project, we use the small version of FMA (Free Music Archive) dataset for the task of music genre classification on 8 genres. The FMA is a large and diverse dataset of annotated music recordings, spanning multiple genres and providing various metadata information. The stratified split of the dataset is already done and it can be found in the metadata. In this way the data is divided into training, validation and testing sets such that the proportion of class labels is the same in each set as it is in the original dataset. This is important to ensure that all the sets have a similar overall class distribution. The FMA small dataset comprises 8000 tracks from a multitude of different artists. The way to split data into train/validation/test is already provided by the authors and it is 8/1/1.

V. Learning and Model Framework

A. ***Architectures***

Given the prevalence of established architectures for music datasets such as GTZAN, we opted to both utilize existing implementations and develop our own original architectures, inspired by the successful models. The following provides a brief overview of these architectures:

1) ***1D Architectures****:*

1) ResNet1D: 1D ResNets typically consist of multiple blocks of 1D convolutional layers and pooling layers, with the residual connections being added between the blocks. The convolutional layers are used to extract features from the input data, while the pooling layers are used to reduce the spatial dimensions and control the complexity of the model. In this context, we implement the 1D ResNet proposed by Allamy and Koerich [7].

2) ***2D Architectures****:*

1) Simple CNN: This is the simplest CNN we created for spectro gram-based classification which only has 25472 parameters. It is composed of four convolutional blocks. The first convolutional layer has a kernel size of 5x5, while the other layers all have the same kernel size of 3x3. Stride and padding are set to 2x2 for the first block, while the other blocks use 2x2 and 1x1 sizes for stride and padding respectively. Furthermore, we use batch normalisation and ReLU as the activation function for each layer. The result is passed to an adaptive average pooling layer before reaching the final linear layer outputting the predicted genre.

2) Simple VGG: We employed a straightforward VGG architecture consisting of four Convolutional layers, each equipped with a 3x3 filter. As a final layer we have a simple linear layer that outputs the predicted class. This architecture served as a means of testing our data and, due to its simplicity, we did not anticipate exceptional training accuracy. We employed VGG16 for its simplicity in transfer learning. Initially, we created our own version of VGG16 and trained it on the GTZAN dataset. After successful training, we saved the model and its weights to use as a pre-trained model for the FMA dataset.

3) nnet1 and nnet2: In this project, we sought to examine the performance of the nnet1 and nnet2 architectures [3] on the FMA dataset. As outlined in the referenced paper, the nnet1 architecture consists of three convolutional layers, each with a 4x4 filter size, followed by a max-pooling layer of size 2x1 (except for the final layer, which utilizes a combination of max and average pooling of size 26x1). The final three layers are dense layers with 300, 150, and 10 hidden units, respectively, serving as the classifier that automatically assigns the input audio to various genres. The output of the final layer represents the probabilities of each genre classification.
Additionally, we employed nnet2, a residual form of nnet1, which features a single residual block of three convolutional layers, each with the same filter size as nnet1. Similar to nnet1, we integrated a combination of max and average pooling, this time with a size of 125x1. To accelerate the training process and enhance the robustness of the final model, we employed batch normalization (BN) . The BN operation reduces internal covariate shift, enabling us to use a significantly larger learning rate. This architecture also includes three dense layers, with 300, 150, and 10 hidden units respectively.

4) ResNet34:
We utilized ResNet34 in two approaches. Initially, we utilized a ready-to-use ResNet34 model from PyTorch, but found that it was lacking in regularization and customization options. As a result, we created our own version of ResNet34 with slight modifications. However, for the purpose of comparison with ResNet18, we employed the ready-to-use ResNet34 model from PyTorch.

5) ResNet18:
We implemented ResNet18 to evaluate its performance in comparison to ResNet34 and determine if a deeper network would yield better results in our scenario.

*3) **SoundFusion Architectures:*** For the 1D-CNN we use the 1D ResNet and we rescale it by decreasing the number of filters of the first residual block from 128 to 16. In this case, the final layer has 64 outputs.

On the other hand, after running multiple experiments with different 2D-CNNs, we noticed that ResNet34 gets the best results. So, we decide to use it by rescaling it to the same number of 16 filters for the first residual block. When it is incorporated in SoundFusion, we add an extra fully connected layer to map the number of outputs from 3136 to 64.

We implement late fusion in two different ways: by feature concatenation and decision weighting [13]. In the first approach, the outputs of the two subnetworks are concatenated and softmax is applied afterwards to find the probability distribution. In decision weighting you need to first calculate the probability distributions from each input modality and then weight these probabilities to obtain the final probability distribution of the music category. We consider that both representations (waveforms and spectrograms) are equally important, therefore the weights are equal.

*B. **Optimization***

All of the baselines are trained with Adam optimizer and a learning rate of 0.0001, except for the Simple CNN which uses a learning rate of 0.001. For the majority of the networks, the mini-batch size is set to 32, except in cases where we are running out of memory and a batch of 16 is used instead (for the large size versions of ResNet). For the smallest network with fewer parameters we were able to experiment with larger batch sizes of up to 256. Additionally for this network, given its simplicity and parameter scarcity, we also experimented on a reduced version of our dataset. Data augmentation parameters are optimized via gridsearch. For the waveform augmentation we add Gaussian noise and amplify or reduce the volume randomly. We augment the spectrograms using both Frequency masking and Time masking [18], which consists in randomly replacing certain parts of the spectrogram along the frequency and time axis with the mean value. This is done using the built-in functions from torchaudio. We also assign a given probability for our data to be augmented and experiment on different values of it, as well as different amounts of masks and their sizes.

VI. RESULTS

A. Regularization techniques

To avoid overfitting we augment the data and add dropout layers. We augment the data for both waveforms and spectrograms:

- **Waveform augmentation** : the best augmentation performances are obtained by adding Gaussian noise with a standard deviation of 0.02 and modifying the volume by a random amplitude factor which is calculated using the following formula: $10^{\mathrm{gain}/20}$, where the gain is is randomly chosen between -12 dB and 12dB.
- **Spectrogram augmentation** : The results of the gridsearch show that the best hyperparameters are the following : an augmentation probability of 1, the maximum mask percentage of 0.3, which controls the maximum possible size that a mask can take, and the number of frequency and time masks to be both 2. Moreover, the results of the gridsearch showed that data augmentation

model	regularization	validation	test
ResNet 1D (70k param.)	N/A	0.472	0.380
	dropout	0.515	0.413
	dropout + augmentation	0.507	0.418
ResNet 2D (445k param.)	N/A	0.523	0.485
	dropout	0.520	0.489
	dropout + augmentation	0.531	0.528

TABLE 1: Results of ResNet 1D and 2D CNNs with and without optimizations. The small size version of the networks is used.

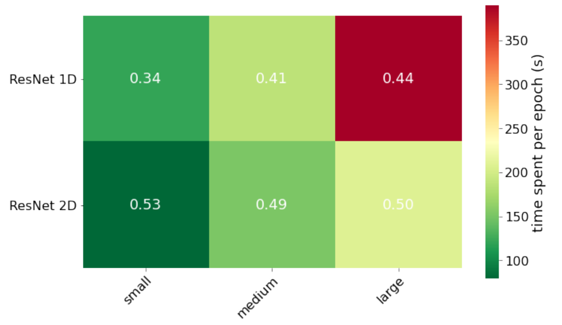


Fig. 1: Computational time heatmap showing the test accuracy (without aggregation) as a function of the number of parameters

positively impacted the accuracy in most cases, as expected, although it was also shown that in some cases it had a negative effect. In particular, in all cases when increasing the number of frequency and time masks to 3, the network performed sub-optimally, regardless of the other parameters. We also noted that by increasing the augmentation probability the network performed better, indicating that our models are prone to overfitting, which is to be expected as we are dealing with a dataset limited in size.

B. Width scaling

Here we present how the performance modifies when using architectures with different number of parameters obtained by width scaling. Because we are experimenting with FMA small, we would expect that larger models would reach overfitting faster. In the Fig. 1 we provide the test accuracy and the computational time per epoch with respect to the number of parameters. For each architecture we vary the number of filters from 16 (small) to 64 (large). The number of parameters for each model is as follows,

1) *ResNet1D*: small - 70k, medium - 280k, large - 1.1m
2) *ResNet2D*: small - 445k, medium - 1.7m, large - 6.8m

C. SoundFusion results

We ran the proposed architecture by using feature concatenation (SoundFusion-1) and, also, by weighted decision (SoundFusion-2). The summary of the results is presented in Table 2, where we show the test accuracy on segments and the one given by aggregation using the majority class. In the same table we provide the performances of the baselines we have implemented. It seems that SoundFusion-1 slightly outperforms ResNet2D-s, the small rescaled version of ResNet2D. Overall, we notice that aggregation increases the accuracy by 2-4%.

Input	Model	Test accuracy	Aggregation
1D	ResNet1D-m [7]	0.406	0.426
2D	pre-trained VGG16	0.492	0.526
	ResNet2D-s [1]	0.528	0.567
	nnet1 [3]	0.344	0.366
	nnet2 [3]	0.475	0.493
	Simple CNN w/o segmentation	0.484	n/a
Mix	**SoundFusion-1**	0.535	0.571
	SoundFusion-2	0.483	0.506

TABLE 2: The best scores obtained with 1D/2D input baselines and SoundFusion multimodal networks.

In Figure 2, we plot the confusion matrices for SoundFusion-1, ResNet1D and ResNet2D. From the confusion matrices, we can observe that the SoundFusion-1 model has the highest accuracy for the HipHop genre with a rate of 77 percent, demonstrating its effectiveness for this specific genre. The ResNet1D model, on the other hand, showed exceptional performance for the experimental genre with an accuracy of 83 percent. The ResNet34 model, despite having a more general approach, still showed good performance for all genres with an overall accuracy of almost 57 percent. These results indicate the potential of each model for specific genres and the importance of choosing the appropriate model for a given task.

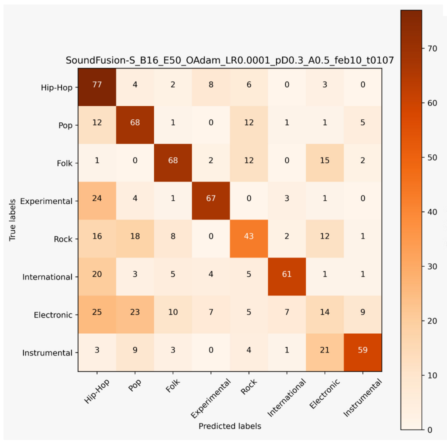
VII. Concluding Remarks

In this study, regularization techniques and width scaling were applied to improve the performance of ResNet 1D and 2D CNNs for a limited data scenario. The results showed that data augmentation and dropout layers were effective in avoiding overfitting, and the proposed SoundFusion architecture outperformed the baseline models.

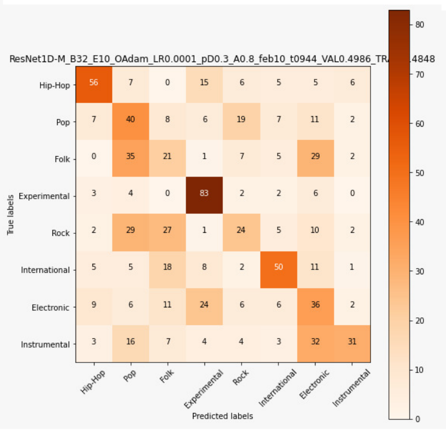
The results of this study demonstrate the effectiveness of regularization techniques and width scaling in improving the performance of CNNs in a limited data scenario. The findings highlight the importance of combining different feature representations and decision making for improved performance. The study also sheds light on the trade-off between model size and overfitting.

To extend this work, future studies could explore different feature extraction methods, as well as other regularization techniques and decision making methods. Additionally, the study could be extended by using larger datasets to validate the results and further improve the performance of the proposed models. Another area of future work could be to fine-tune

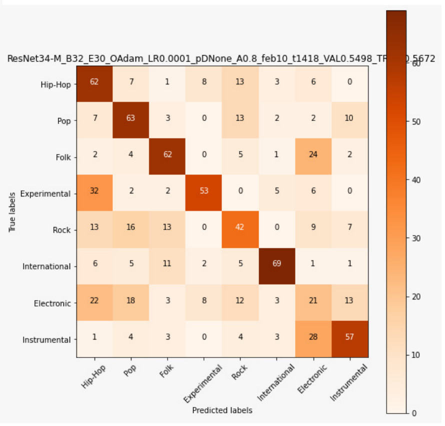
Fig. 2: Confusion matrices



(a) SoundFusion-1



(b) ResNet1D



(c) ResNet2D

the training hyperparameters of SoundFusion such as the batch size, optimizer, learning rate, augmentation probability, number of residual or dropout layers.

Throughout the project, the authors have gained valuable experience in building neural networks from scratch, concatenating layers, and avoiding common mistakes such as forgetting to use .eval() or applying the sigmoid when using categorical cross entropy. The authors have also learned the importance of careful experimentation and proper implementation of regularization techniques to prevent overfitting.

REFERENCES

[1] S. R. Kaiming He, Xiangyu Zhang and J. Sun, “Deep residual learning for image recognition,” arXiv, 2015.

[2] D. S. Kamalesh Palanisamy and A. Yao, “Rethinking cnn models for audio classification,” arXiv, 2020.

[3] W. Zhang, W. Lei, X. Xu, and X. Xing, “Improved Music Genre Classification with Convolutional Neural Networks,” pp. 3304–3308, 2016.

[4] T. L. H. Li, A. B. Chan, and A. H. W. Chun, “Automatic musical pattern feature extraction using convolutional neural network,” 2010.

[5] S. Sigtia and S. Dixon, “Improved music feature learning with deep neural networks,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6959–6963, 2014.

[6] K. Choi, G. Fazekas, M. B. Sandler, and K. Cho, “Convolutional recurrent neural networks for music classification,” *CoRR*, vol. abs/1609.04243, 2016.

[7] S. Allamy and A. L. Koerich, “1d cnn architectures for music genre classification,” 2021.

[8] S. Dieleman and B. Schrauwen, “End-to-end learning for music audio,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6964–6968, 2014.

[9] J. Pons, O. Nieto, M. Prockup, E. M. Schmidt, A. F. Ehmann, and X. Serra, “End-to-end learning for music audio tagging at scale,” *CoRR*, vol. abs/1711.02520, 2017.

[10] a. J. K. J. Lee, “Fine-tuning convolutional neural networks for music genre classification,” arXiv, 2011.

[11] A. K. A. J. Stowell, D. Stowell and M. D. Plumbley, “End-to-end deep learning for audio-based bird species recognition,” Scientific Reports, 2018.

[12] K. Gadzicki, R. Khamsehashari, and C. Zetzsche, “Early vs late fusion in multimodal convolutional neural networks,” pp. 1–6, 2020.

[13] Y. Li, Z. Zhang, H. Ding, and L. Chang, “Music genre classification based on fusing audio and lyric information,” *Multimedia Tools and Applications*, 12 2022.

[14] K. W. Cheuk, K. Agres, and D. Herremans, “The impact of audio input representations on neural network based music transcription,” 2020.

[15] Y.-H. Cheng and C.-N. Kuo, “Machine Learning for Music Genre Classification Using Visual Mel Spectrum,” *Mathematics*, vol. 10, pp. 1–19, November 2022.

[16] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, “Fma: A dataset for music analysis,” 2016.

[17] A. Elbir, H. Ilhan, G. Serbes, and N. Aydin, “Short time fourier transform based music genre classification,” pp. 1–4, 04 2018.

[18] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, “SpecAugment: A simple data augmentation method for automatic speech recognition,” sep 2019.