Audio Retro Style Transfer and Recommendation System Based on Genre Identification

Shuodi Yin
23006440
14/03/2024

Introduction

Led by the success of neural style transfer in visual arts, there has recently been a growing trend in the effort of music style transfer (Dai S, 2018). The challenge lies in the complexity of audio with its multi-layered characteristics, distinct from image representations, involving complex computer audio processing and AI model learning. This project proposes an innovative approach that integrates the concept of image style transfer with advanced audio models to explore and create entirely new music styles. By developing a custom Convolutional Neural Network (CNN) model, this project achieves vintage style transfer for user-uploaded audio, utilizing audio generated by the RAVE model as the style input. The effectiveness of the model is evaluated by comparison with direct processing audio from RAVE and is continually improved through parameter adjustments and optimization strategies. Moreover, incorporating an M5 model trained on genre recognition from an audio dataset, this project also develops a recommendation system, showcased through a Graphical User Interface (GUI). This exploration not only paves new paths in audio style transfer technology but also deepens the understanding of audio characteristics, providing valuable insights for the future of audio processing technology.

Background

Inspired by Alish Dipani's (2018) "Neural Style Transfer," this project explores audio style transfer, transforming audio signals into Short-Time Fourier Transform (STFT) spectra and processing them with a CNN model tailored for audio data. The model, featuring a one-dimensional convolutional layer with 1025 input and 4096 output channels, employs the Gram matrix and style loss for effective style transfer. Utilizing the Adam optimizer and Griffin-Lim algorithm, it efficiently reconstructs audio, merging content and style features, and enhances performance through optimized audio preprocessing and efficient spectrum and audio file handling.

Furthermore, the project advances genre recognition by training an M5 model on a new dataset, incorporating Olteanu's (2021) insights on audio feature visualization and song recommendation into a user-friendly interface for interactive music exploration. This method not only streamlines and innovates in audio modeling but also broadens research into audio feature and similarity analysis, establishing a new paradigm in audio style transfer and demonstrating theoretical concepts' practical applications.

Method

Initially, I utilized the vintage version of the RAVE pre-trained model to generate audio and analyze its features. To match the duration of the target audio and facilitate style transfer, I configured the latent space dimensions and adjusted sampling rate. This process involved creating specific directories for storing the model and the synthesized audio. Following this, I downloaded the vintage model and stored it in the designated directory. After setting up the environment, I performed latent space interpolation using the model to produce coherent audio segments, which were then combined into a single file. I employed "librosa" to trim silence and extract audio features such as the Mel spectrogram, spectral centroid, and chromagram, and used "matplotlib" to visualize these features before saving the audio.

Using the generated audio as style input, I initiate the critical phase of my project: audio style transfer. I deploy a custom CNN model that leverages "self.cnn1" to process input audio, yielding a one-dimensional vector. This model uses a Gram Matrix and Style Loss classes to capture the essence of the audio style, with the Gram Matrix analyzing feature relationships and the Style Loss utilizing mean squared error to quantify the differences between style and content audio, facilitating the style transfer. The pivotal element is a sequential model that intertwines convolutional layers for feature extraction with style loss layers that compute and optimize the loss throughout the transfer process. To maintain the integrity of the original model's parameters, I employ a deep copy technique. I utilize the "get_style_model_and_losses" and "get_input_param_optimizer" functions to obtain the model, assess style losses, and fine-tune the optimizer. This approach allows me to tweak the parameters, reducing loss and nudging the content audio closer to the target style. Preprocessing is crucial to ensure the audio is compatible with the PyTorch framework, with "librosa" standardizing sampling rates, and reshaping the spectrum. The style transfer comes to life through the "run_style_transfer" function, a multistep procedure involving model and loss acquisition, algorithm execution, and post-processing to realize the final styled audio output.

Next, I processed the original audio with the RAVE model for direct comparison against the style-transferred audio, using analyses of spectral centroids, Mel spectrograms, and chromagrams to gauge the style transfer's effectiveness. Following that, I implemented the M5 CNN architecture for genre classification. This required uniform resampling of audio to a consistent sampling rate. Audio files were divided into training and validation sets. The M5 model, consisting of convolutional, pooling, and fully connected layers, leveraged cross-entropy loss and the Adam optimizer for training. Over several epochs, the model's parameters were refined and losses measured, with the best parameters saved upon achieving the lowest validation loss. Matplotlib was used to visualize the training and validation losses. Post-training, the model underwent evaluation on a dataset after necessary preprocessing like resampling and channel adjustment.

Moreover, the project integrates a music genre classification and recommendation system in a GUI, utilizing the M5 model alongside a K-Nearest Neighbors classifier. "torchaudio" is employed for audio loading and preprocessing, ensuring compatibility with model inputs. The GUI includes "preprocess_audio" for input preparation and "classify_audio" for genre prediction. A K-Nearest Neighbors classifier, trained with cosine similarity for song recommendations, is incorporated. The interface, built with "Tkinter", facilitates audio file selection, classification, recommendations, and displays the results for user interaction.

Results

During the vintage audio generation phase, I used the RAVE model's vintage version to create a 30-second vintage-style audio piece, analyzing its feature map to understand the specific style's characteristics.

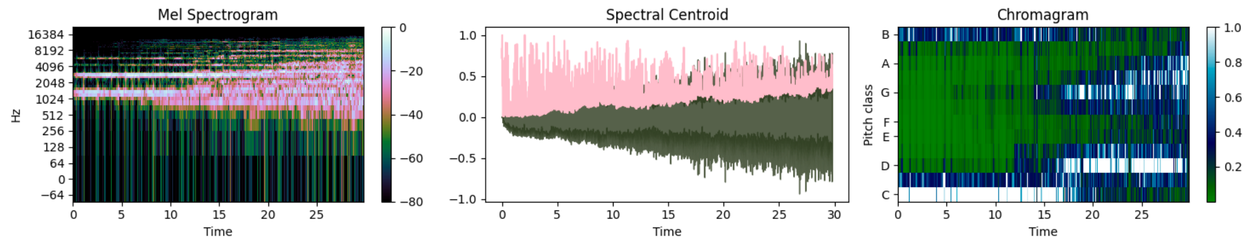


Figure 1: RAVE Vintage Audio Feature Map

The Mel spectrogram shows the audio beginning at a lower volume and significantly increasing towards the end. Waveform and spectral centroid graphs indicate a concentrated rise in spectral energy in the second half. The chromagram shows the audio starting with lower pitches, with an increase and diversification in pitch as it progresses. In summary, the generated audio demonstrates a progressive change from low to high.

After generating, I executed style transfer between this and user-uploaded content audio, experimenting with different learning rates, and loss function weights, and employing various loss functions and optimizers. Comparative results feature Figure 2, produced with Mean Squared Error (MSE) and Adam optimizer, and Figure 3, utilizing Mean Absolute Error (MAE) with Stochastic Gradient Descent (SGD) optimizer:

	lr	style_weight	style loss
The loss function is MSE The optimizer is Adam	0.01	2500	no change stable at 0.012508
	0.05		slightly decreased from 0.012654 to 0.012653
	0.1		**slightly decreased from 0.011750 to 0.011749**
	0.01	5000	no change stable at 0.049959
	0.05		slightly decreased from 0.048213 to 0.048211
	0.1		slightly decreased from 0.012177 to 0.012176

Figure 2: Style Loss Comparison with Varying Learning Rates and Weights under MSE Loss and Adam Optimizer

	lr	style_weight	style loss
The loss function is MAE The optimizer is SGD	0.01	2500	no change stabilizing at 0.071844
	0.05		**no change stabilizing at 0.068326**
	0.1		no change stabilizing at 0.071865
	0.01	5000	no change stabilizing at 0.139341
	0.05		no change stabilizing at 0.146549
	0.1		no change stabilizing at 0.141471

Figure 3: Style Loss Comparison with Varying Learning Rates and Weights under MAE Loss and SGD Optimizer

In Figure 2, varying learning rates and style weight settings lead to a slight decrease or stable style loss. Notably, a learning rate of 0.1 and style weight of 2500 achieve the lowest style loss at 0.011749. Using the SGD optimizer, the style loss remains unchanged across different combinations of learning rates and style weights. Specifically, when the style weight is 2500 and the learning rate is 0.05, the style loss stabilizes at a lower value of 0.068761. These findings elucidate the impact of parameter adjustments on style loss performance, aiding in optimizing neural network parameters. The optimizer choice significantly affects loss stability and reduction. Data indicate that the Adam optimizer with MSE minimizes style loss more consistently across various settings. Conversely, the SGD optimizer, despite its stability in certain configurations, does not significantly reduce style loss. Hence, choosing the right optimizer and loss function is key for lower style loss in audio style transfer.

I plan to compare features among the original audio, the audio processed by the vintage RAVE model, and the audio from the style transfer. By analyzing spectral centroids, Mel spectrograms, and chromagram, I aim to evaluate the effectiveness of the style transfer model.

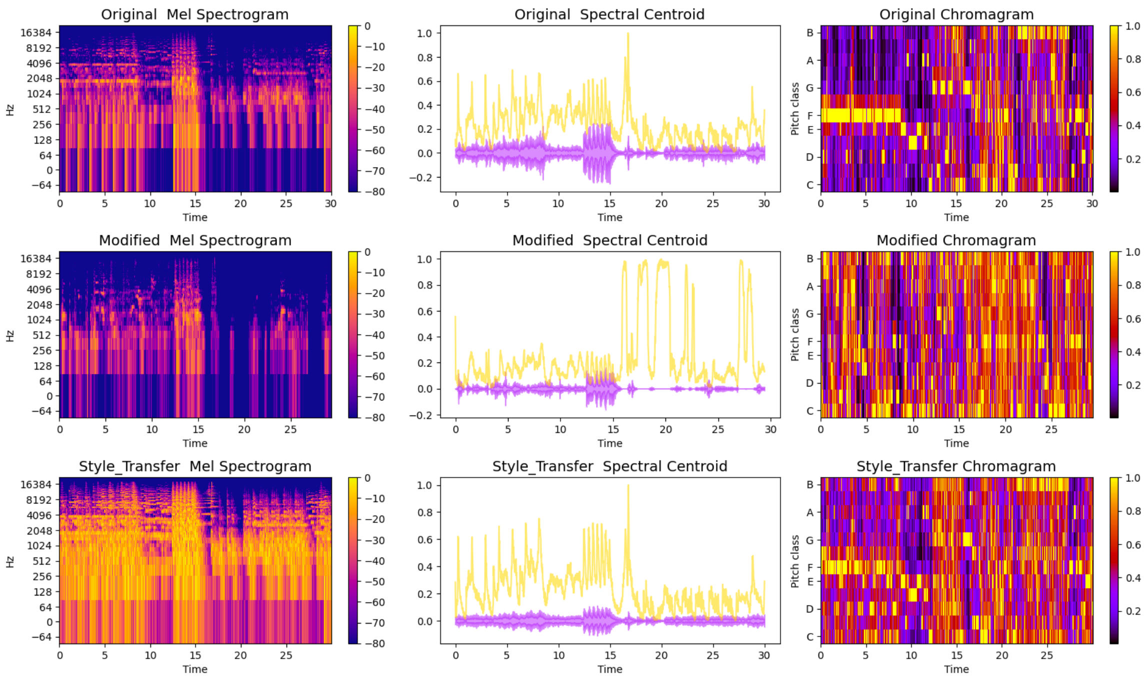


Figure 4: Comparison of Audio Features: Original vs. Vintage-Processed vs. Style-Transferred Audio

From this visualization, we can see the features of three different audio files:

- The Mel Spectrogram, mimicking human auditory perception, shows that the "Original " and "Modified "have similar energy components, suggesting the vintage processing retains core spectral traits of the original. Conversely, "Style_Transfer " audio exhibits a distinct spectral structure with enhanced audio intensity, indicating the style transfer not only captures vintage timbre but also infuses creativity, potentially altering music perception and emotional nuances.
- The Spectral Centroid, indicating an audio signal's brightness or pitch, reveals the "Modified " audio occasionally reaches higher spectral centroids, hinting at melodic enhancements. "Original " and "Style_Transfer " audios display more consistency, suggesting that the style transfer may have been based on the original audio to reduce the music's dynamic range, resulting in a more uniform timbre.

- The Chromagram represents the intensity of the 12 different pitch classes in the musical signal. The "Modified " displays more yellow and orange areas compared to the "Original, " indicating an enhancement in pitch activity. This suggests that RAVE model processing emphasizes certain pitches and enriches the harmony, showcasing a vintage effect. The "Style_Transfer " shows a more uniform distribution of harmonic activity, without pronounced bright spots, potentially signifying that the style transfer introduced new harmonic structures or adjusted the distribution of harmony.

These insights indicate that RAVE model processing tends to amplify and broaden existing harmonics rather than altering music's fundamental structure. The style transfer, however, introduces novel musical elements and expressions, enriching the original content with new styles and harmonic progressions. This could entail scale recombination, harmony recomposition, and alterations in melody and rhythm.

Then, I trained a genre recognition model using the GTZAN Dataset and assessed its performance by visualizing training and validation losses.

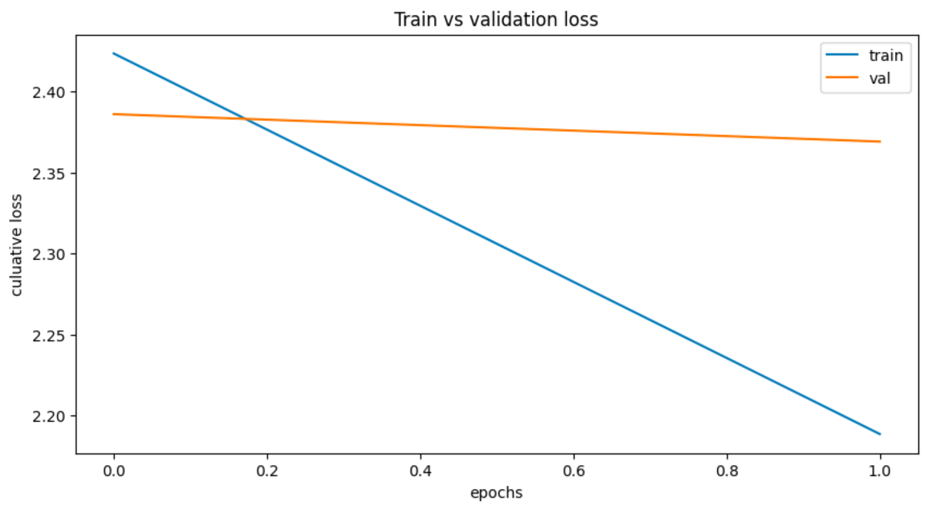


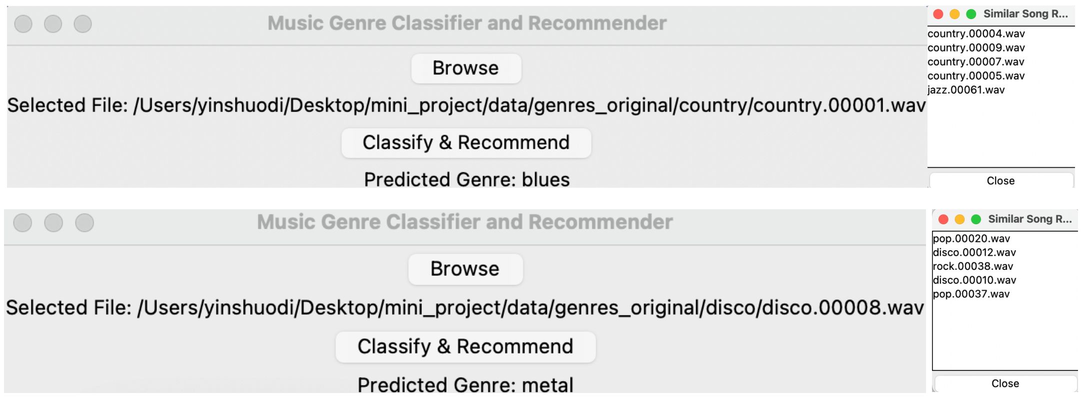
Figure 5: Train vs validation loss

This graph illustrates the neural network's training and validation losses throughout its training phase, allowing us to draw several conclusions:

- Loss Trends: A notable decrease in training loss (blue line) with each epoch suggests effective learning and fitting to the training data.
- Validation Loss Behavior: A relatively stable validation loss (orange line) may indicate overfitting to the training data, as ideal conditions would show both losses decreasing and the validation loss staying close to the training loss.

In essence, the graph indicates that while the model improves on the training set, this improvement doesn't extend to the validation set, hinting at potential underperformance with new data. An initial test successfully identified an audio file as belonging to the blues genre. However, subsequent testing revealed occasional incorrect categorization of audios into a single genre, indicating the model's overfitting, which leads to unstable predictions across different genres.

Finally, I've developed a straightforward music recommendation system, combining a CNN-based genre recognition system with song similarity analysis via cosine similarity, all accessible through a GUI. Users upload a track, receive its genre, and the names of the top 5 similar songs.


Music Genre Classifier and Recommender

Browse

Selected File: /Users/yinshuodi/Desktop/mini_project/data/genres_original/country/country.00001.wav

Classify & Recommend

Predicted Genre: blues

Similar Song R...

country.00004.wav
country.00009.wav
country.00007.wav
country.00005.wav
jazz.00061.wav

Close

Music Genre Classifier and Recommender

Browse

Selected File: /Users/yinshuodi/Desktop/mini_project/data/genres_original/disco/disco.00008.wav

Classify & Recommend

Predicted Genre: metal

Similar Song R...

pop.00020.wav
disco.00012.wav
rock.00038.wav
disco.00010.wav
pop.00037.wav

Close

Figure 6: GUI Interface Example

Discussion

This project delved into audio style transfer and genre identification, showcasing style transfer's potential in audio. Leveraging the RAVE model, we managed to both preserve the original audio's essence and introduce new musical elements, highlighting the technique's capability to generate new musical styles. The combination of MSE with the Adam optimizer improved style loss stability, indicating a good fit with our dataset's features. Conversely, the SGD optimizer and MAE loss function were less effective, suggesting a misfit with audio style transfer needs. The genre identification model faced overfitting, affecting genre recognition accuracy, likely from dataset limitations on feature variety. Future efforts will focus on enriching the dataset with more diverse audio to boost model generalization and exploring advanced neural networks like RNNs and Transformers for performance improvement. Despite challenges in genre identification, the project's achievements in audio style transfer highlight deep learning's significant promise in music innovation and exploration.

Conclusion

In this project, we embarked on an exploration of audio style transfer and genre recognition, leveraging the RAVE model and CNNs to create and analyze retro-styled audio. Through detailed analysis, we enhanced the visual understanding of audio characteristics and also innovated in audio preprocessing and spectrum analysis. We also developed a genre-based recommendation system with a user-friendly interface, despite facing challenges like overfitting in genre classification. These obstacles have deepened our understanding of deep learning model generalization and offered insights for future improvements in model design and data handling.

Ethical considerations

we must consider the implications of dataset usage and model applications on artists' creative rights, copyright laws, and user privacy, ensuring compliance with copyrights and respect for stakeholders' interests.

LLM disclaimer

I utilized large language models (LLMs) as an auxiliary tool, which assisted me in fixing errors in the code, as well as in translating the essay writing.

Word count:1990

Bibliography

[1] Anagha, J., Jiaaro and Murth, V. (2018) How to split the audio file in Python, Stack Overflow. Available at: https://stackoverflow.com/questions/43204441/how-to-split-the-audio-file-in-python (Accessed: 14 March 2024).

[2] AnBe and Bryan Oakley (2017) How to return the selected items of a listbox when using wait_window() in Tkinter, Stack Overflow. Available at: https://stackoverflow.com/questions/37219191/how-to-return-the-selected-items-of-a-listbox-when-using-wait-window-in-tkinte (Accessed: 09 March 2024).

[3]Damicelli, F. (2023) PyTorch DataLoader: Understand and implement a custom collate function. Available at: https://fabridamicelli.github.io/posts/2023-09-13-pytorch-dataloader-collate.html (Accessed: 14 March 2024).

[4] Dai, S., Zhang, Z. & Xia, G.G. (2018) Music style transfer: A position paper. *arXiv preprint arXiv*, 18(03), pp.8-41.

[5]Dipani, A. (2018) Neural Style Transfer Audio, GitHub. Available at: https://github.com/alishdipani/Neural-Style-Transfer-Audio/blob/master/NeuralStyleTransfer.py (Accessed: 08 March 2024).

[6] Hamza (2019) How to access a desired path with filedialog.askopenfilename() in Tkinter, Stack Overflow. Available at: https://stackoverflow.com/questions/54785138/how-to-access-a-desired-path-with-filedialog-askopenfilename-in-tkinter (Accessed: 09 March 2024).

[7] Kenan (2022) Torchaudio loads audio with a specific sampling rate, Stack Overflow. Available at: https://stackoverflow.com/questions/71108331/torchaudio-load-audio-with-specific-sampling-rate (Accessed: 09 March 2024).

[8] Lheureux, A. (2022) Music genre classification using Librosa and Tensorflow/Keras, Paperspace. Available at: https://blog.paperspace.com/music-genre-classification-using-librosa-and-pytorch/ (Accessed: 11 March 2024).

[9] Mcumcu and OysterShucke (2021) Tkinter widget.cget('variable'), Stack Overflow. Available at: https://stackoverflow.com/questions/63871376/tkinter-widget-cget-variable (Accessed: 09 March 2024).

[10] Numpy (2022) numpy.expand_dims. Available at: https://numpy.org/doc/stable/reference/generated/numpy.expand_dims.html (Accessed: 14 March 2024).

[11] Olteanu, A. (2020) Work W/ audio data: Visualise, classify, recommend, Kaggle. Available at: https://www.kaggle.com/code/andradaolteanu/work-w-audio-data-visualise-classify-recommend/notebook#Introduction (Accessed: 08 March 2024).

[12] Opidi, A. (2023) Pytorch loss functions: The ultimate guide, neptune.ai. Available at: https://neptune.ai/blog/pytorch-loss-functions#Mean-Absolute-Error (Accessed: 09 March 2024).

[13]Pandas (2024) pandas.Series.to_dict - pandas 2.2.1 documentation. Available at: https://pandas.pydata.org/docs/reference/api/pandas.Series.to_dict.html (Accessed: 14 March 2024).

[14] Seberino (2023) Question about the use of a torch.max function to calculate accuracy, PyTorch Forums. Available at: https://discuss.pytorch.org/t/question-about-use-of-torch-max-function-to-calculate-accuracy/187500 (Accessed: 09 March 2024).

[15] Sklearn (2007) scikit.preprocessing.LabelEncoder. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html (Accessed: 09 March 2024).

[16] Team, I. A. (no date) Rave models download page, RAVE models. Available at: https://acids-ircam.github.io/rave_models_download (Accessed: 09 March 2024).

[17] Torch (no date) torch.optim - PyTorch 2.2 documentation. Available at: https://pytorch.org/docs/stable/optim.html (Accessed: 09 March 2024).

[18] Tuong Tran Ngoc (2023) Everything you need to know about pytorch's collate_fn. Available at: https://tuongtranngoc.github.io/posts/2023/collate_fn/ (Accessed: 14 March 2024).