Img2Music

Bryan Carlo Miguel
bryancarlomiguel@gmail.com
Concordia University
Montreal, Quebec, Canada

Abstract

This project aims to bridge the gap between image and audio by generating short music snippets given an album cover image. In this project, we develop *Img2Music*, a model consisting of two parts: a classifier used to predict the genre of the album cover, and a set of variational autoencoders (VAEs) used to generate genre-sepecific music snippets. *Img2Music* is able to generate coherent and recognizable music snippets with a general genre prediction accuracy of 60.75%.

1 Introduction

Although there are plenty of models that generate some form of audio, there isn't anything that generates coherent music snippets from images. Previous works such as Google's Deepmind and fffiloni's tool appear to do the same, but these appear to be closed-source and use a text prompt as part of the conditioning signal for the generative model. My approach differs by strictly using the output feature vector of the CNN as the conditioning signal.

The goal of this project is to bridge the gap between images and audio by developing a network that generates music based on what it 'sees'. This could make viewing images a more immersive experience. A potential application would be drafting album cover images and seeing what type of music captures the 'vibe' or 'style' of that image.

This report presents *Img2Music*, a model that generates coherent music snippets from the features of an album cover image. It consists of two main parts: a classifier and an audio generator. First, the album cover's features are used to predict what genre it would most likely be in. This predicted genre and image features are passed into the audio generator, which samples from a *Variational Autoencoder* (VAE) to obtain generated audio.

The main advantage of this approach is in its simplicity. There is no need to incorporate text-based conditioning signals as in other approaches. This reduces the training burden, as well as simplifying our model architecture. However, because of that, the main challenge is to be able to produce coherent and passable generated music given the complexity and training cost constraints.

2 Method

For simplicity, the genres that I am using are: Classical, HipHop, Pop, and Jazz. These were selected because these genres are some of the most representative in music. Furthermore, these genres are sufficiently different from each other such that the embeddings for each genre - for both images and audio - are separable between genres.

2.1 Classifier

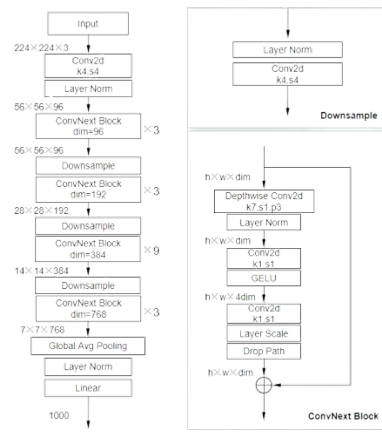

Figure 1: ConvNeXT model architecture.

We are aiming for a classification model that is able to accurately predict an album cover's genre whilst also being simple enough such that it is relatively simple to train. First, I experimented with a hand-rolled VGG16 netowrk using pre-trained weights provided by PyTorch. I found that overall accuracy would not increase past around 45% even after increasing the number of convolutional layers. After, I decided to experiment with models featuring residual connections. I experimented with ResNet, but found that ConvNeXT (Figure 1) offers better performance while being as similar to train as the aforementioned networks.

Further optimizations to the training pipeline included experimenting with dataset augmentations to improve generalization capabilities, tuning learning rate and weight-decay values, and experimenting with LR schedulers. More details in the *Experimental Results* section.

2.2 Audio Generation

Similar to the classification model, we are aiming for a network that is able to generate coherent music snippets of a specified genre that is relatively easy to train, Although I first started experimenting with audio diffusion models, I quickly realized that they were too complex and computationally expensive. As such, I opted for Variational Autoencoders (VAEs) instead.

From experimentation, I observed that training a singular VAE on the entirety of the dataset across different genres resulted in sub-par performance and generalization. This is most likely due to

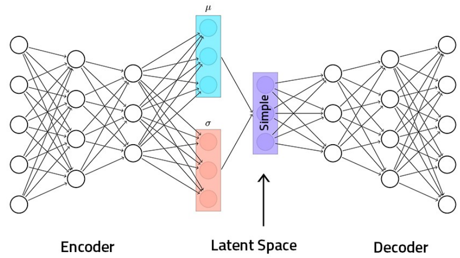

Figure 2: Simple VAE model architecture.

the complex sequential patterns present in music. For VAEs, these patterns are likely very difficult to differentiate which genre they belong to, which means that the network cannot learn the distribution function for each genre.

The workaround to the above problem is to separately train a VAE for each genre. This results in a network that can effectively learn the distribution function of music samples within a genre. This approach allows us to generate audio whilst keeping the training pipeline and model relatively simple.

Incorporating Image Features The genre-specific VAEs mentioned above still suffered from poor generalization, with decoded latent vector samples being barely coherent and recognizable. The approach I took was to perform *latent code perturbation*. Essentially, we would take a known embedding and add our image's feature vector to it, resulting in a new latent space vector to use for sampling. Qualitatively, the results are a more stable, coherent, and recognizable audio sample that differs slightly from the original sample. We can model the sampled embedding vector $\hat{l}$ as $\hat{l} = l + \alpha\hat{v}$, where l is the embedding of a known sample, $\hat{v}$ is the feature vector of the input image, and α controls how much of the feature vector to include in the embedded sampling vector.

2.3 Combining Everything

A single forward pass of obtaining a music sample from an input image is summarized in Algorithm 1.

Algorithm 1: Single forward pass in Img2Music.

Input: An input image v
1. Pass v into the classifier to obtain predicted genre $genre_i$;
2. Select VAE_i that corresponds to $genre_i$;
3. Randomly select a known input's embedding vector l;
4. Compute the new embedding sampling vector $\hat{l} = l + \alpha\hat{v}$;
5. Pass $\hat{l}$ into the decoder of VAE_i;

We will be training the two parts separately, and then combine everything into a single model.

3 Experimental Results

3.1 Classifier

Dataset The dataset we use is *20k Album Covers within 20 Genres* (ACG). It contains 1000 of the most popular album covers for each genre. Although not specified, it can be observed in Figure 3 that the dataset encompasses album covers over a wide time-span.

Figure 3: Sample images from ACG limited to our selected genres (classical, hiphop, pop, jazz).
The oldest is Elvis' *Christmas Album* released in 1957 (row 0, col 1), and the most recent is The Weekend's *Starboy* released in 2016 (row 2, col 3).

The images are pre-processed and normalized to be 224×224 tensors. We apply data augmentations - such as Gaussian noise, random flips, crops, and rotations - to help boost the model's generalization capabilities.

Figure 4: Album covers with the following augmentations: random horizontal flip, random rotation, random resize, sample Gaussian noise.

Evaluation Metrics We will be using the standard classification metrics - recall, precision, f1-score, overall accuracy - to measure our model's performance on the test set. Even though we have done our best to reduce the subjectiveness of the problem so far, an album cover can look like it belongs to a different genre compared to the one it actually is. As such, the cost of recommending the wrong genre is arguably equal to the cost of missing the correct one. Because of that, we will be focusing on the *f1-metric*, which balances the weights of false negatives and positives. Additionally, I will be explaining some caveats with the interpretation of the performance results regarding the dataset later once we evaluate the network's performance.

Training Pipeline and Hyperparameters We using a ConvNeXT network intialized with pretrained weights from *Imagenet1k*. The dataset is split with a training/validation/testing split of ratios 0.8/0.1/0.1 respectively. We pass the all training images in batches of 50 per epoch. The model is trained end-to-end with Adam using cross-entropy loss with a learnign rate $\alpha = 0.0001$ and weight decay $\lambda = 0.005$. We use a LR scheduler to lower the learning rate whenever it plateaus, as well as a training stopper that triggers whenever the model's performance on the validation set does not improve over the best validation loss with a patience of $p = 5$.

Experimental Results Evaluating our model on the test set, we obtain the following confusion matrix (Figure 5). The secondary confusion matrix on the right has values logged and normalized to better identify 'hotspots'.

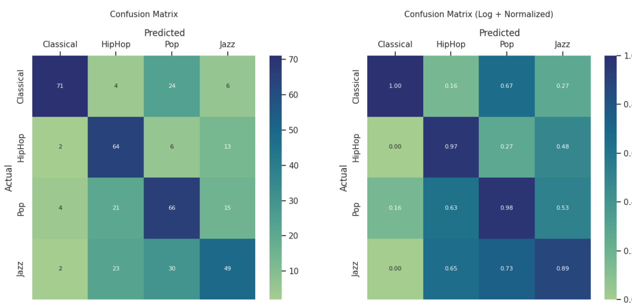

Figure 5: Confusion matrix of the model on the test set.

We can see that, for classical album covers, it was mostly confused for pop album covers. Similarly, hip-hop album covers were mostly confused for jazz album covers, pop album covers were mostly confused for hop-hop and jazz covers, and jazz covers were mostly confused for hip-hop and pop album covers.

Class	Recall (%)	Precision (%)	F1-score (%)
Classical	67.62	67.62	67.62
HipHop	75.29	75.29	75.29
Pop	62.26	62.26	62.26
Jazz	47.12	47.12	47.12

Table 1: Classification metrics obtained from the confusion matrix of the test set. Overall Accuracy: 60.75%

Table 1 contains performance metrics per class obtained from the above confusion matrix. We can see that hip-hop has the highest f1-score at 75.29% and jazz has the lowest f1-score at 47.12%. This suggests that hip-hop contains the most distinct album covers and their features are more separable compared to other genres. This also suggests that jazz is the least distinct and most confused among the other genres.

Interpretation Caveat Album covers are highly subjective in addition to possibly representing multiple genres simultaneously. For example, despite being labeled *HipHop*, the Weekend's *Starboy* (row 2, col 3 in Figure 3) is argued to be *Pop* as much as *HipHop*, if not more. Because of that, despite being relatively low, I find that our overall accuracy of 60.75% to be perfectly reasonable. Potential improvements with regards to this caveat is discussed further in *Potential Improvements*

3.2 VAE

Dataset We use the GTZAN dataset, which provides brief 30 second music snippets for a wide variety of genres. We keep audio as a waveform, with time-domain samples, and we pre-process each sample to be 15 seconds long. We set the sampling rate to 22.05KHz, which results in a vector of size 330,000 per 15 second sample. Since we're working on a reconstruction problem, no data agumentations are performed.

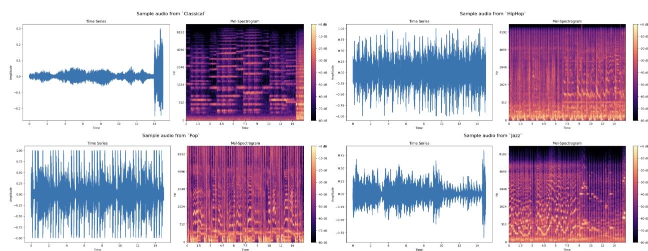

Figure 6: Waveform and mel-spectogram visualizations for each genre. Top left represents *classical*, top right represents *hip-hop*, bottom left represents *pop*, and bottom right represents *jazz*.

Evaluation Metrics Our primary objective evaluation metric is the *Frechet Audio Distance* (FAD). FAD is a measure of how close generated audio is compared to real audio by comparing their latent space representations from a pre-trained model. In other words, it is a measure of how well the generated audio fits into the distribution of the real audio.

The range of FAD is FAD $\in [0, \infty)$. A value of 0 represents a perfect match, and increasing values indicate that dissimilarities exist. We can't use any reference values, since the value itself is problem-specific.

We are using a pretrained VGGish model, whose embeddings capture perceptual and semantic properties of audio such as timbre, pitch, and texture. This allows for a metric that is better able to capture similarity compared to simply comparing the input and output waveform tensors.

Training Pipeline and Hyperparameters We are using a simple VAE inspired from InstructME without any pre-trained weights. The dataset is split with a training/testing split of ratios 0.8/0.2 respectively. We are using a batch size of 1, and the model is trained end-to-end with Adam using mean squared error (MSE) loss with a learning rate of $\alpha = 0.0002$. We train for 500 epochs, without any LR scheduling or early exit mechanism. From eperimentation, I find that anything above 500 epochs yields diminishing returns on performance.

Experimental Results Evaluating the model on the test set, we find mean FAD scores 555.25, 393.48, 485.23, 657.331 for our Classical, HipHop, Pop, and Jazz VAEs respectively, as can be seen in Table 2. HipHop having the lowest score indicates that the reconstructed embeddings match the closest with the embeddings from our real samples the closest on average. This means that generated HipHop tracks have the highest fidelity. The opposite can be said for Jazz output, which has the highest mean FAD score of 657.331.

HipHop having a lower mean FAD score could indicate that its embeddings are more representative of the input data. In other words, HipHop songs were easier to model inside the VAE due to its simple melody and loop-based structure compared to other genres like Classical and Jazz.

Class	$\overline{FAD}$
Classical	555.25
HipHop	393.48
Pop	485.23
Jazz	657.331

Table 2: Mean FAD metrics per class on the test set.

Interpretation Caveat One thing to note is that, qualitatively, the generated audio samples are coherent and resemble the original audio, but degraded with noise. From listening to the audio samples myself, I could not differentiate the levels of degredation between different VAEs. This means that, despite some VAEs having a higher or lower mean FAD metric, end-users and/or untrained listeners may be unable to determine which has objectively better or worse output.

Furthermore, whether these samples are considered 'passable' depend on the generated audio itself. Since we can only provide visualizations in this report, it is recommended to view the showcase file $[img2music]\_Model\_Showcase.ipynb$ to listen to these generated samples.

4 Potential Improvements

4.1 Classifier

Support for multiple genres In this implementation, we take only the most likely genre that an album cover could represent. Due to the subjective nature of album cover images, other genres could be as correct as the predicted one. A potential improvement would be to take the first N most likely genres and then create music snippets for those genres. This could increase the objective performance of the model, as well as improve the quality of outputs by having the user choose which snippet sounds the best to them.

4.2 Audio Generator

Improved Model Architecture This project uses a very simple VAE that is able to balance performance and training cost given the constraints of this project. A more complex architecture, such as a simple diffusion network for music, may produce more coherent and clear music snippets without the distortion problems that our current network suffers from.

5 Conclusion

In this proejct, we developed a model that generates genre-specific audio snippets based on the features of an album cover image. The network maps visual input features to waveform features, enabling cross-modal generation. Despite producing recognizable and coherent audio, both our qualitative and quantitative metrics suggest that there still is room for improvement. This projects demonstrates a good and simple foundation for image-to-audio generation and highlights the problems present specifically in generative music.

References

[1] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," *Proc IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 11976–11986, 2022.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 770–778, 2016.

[3] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint arXiv:1409.1556*, 2015.

[4] B. Han, J. Dai, W. Hao, X. He, D. Guo, J. Chen, Y. Wang, Y. Qian, and X. Song, "InstructME: An Instruction Guided Music Edit and Remix Framework with Latent Diffusion Models," *arXiv preprint arXiv:2308.14360*, 2023.

[5] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[6] C. Doersch, "Tutorial on Variational Autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[7] O. Rybkin, K. Daniilidis, and S. Levine, "Simple and Effective VAE Training with Calibrated Decoders," *arXiv preprint arXiv:2006.13202*, 2020.

[8] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet Audio Distance: A Metric for Evaluating Music Enhancement Algorithms," *arXiv preprint arXiv:1812.08466*, 2018.

[9] M. Kerr, "20k Album Covers within 20 Genres," Kaggle, 2023. [Online]. Available: https://www.kaggle.com/datasets/michaeljkerr/20k-album-covers-within-20-genres

[10] A. Olteanu, "GTZAN Dataset – Music Genre Classification," Kaggle, 2021. [Online]. Available: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification

[11] S. Hershey *et al.*, "CNN architectures for large-scale audio classification," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 131–135. [Online]. Available: https://github.com/tensorflow/models/tree/master/research/audioset

[12] F. Filoni, "image-to-music-v2," Hugging Face Spaces, Apr. 2025. [Online]. Available: https://huggingface.co/spaces/fffiloni/image-to-music-v2. [Accessed: Feb. 12, 2025].

[13] Generative Media Team, "Generating audio for video," Google DeepMind, 17 June 2024. [Online]. Available: https://deepmind.google/discover/blog/generating-audio-for-video/. [Accessed: Feb. 10, 2025].

[14] Y. Wang, Y. Li, and H. Zou, "Masked face recognition system based on attention mechanism," Information, vol. 14, no. 2, art. no. 87, 2023, doi: 10.3390/info14020087.