HIGH SCHOOL EDITION
Journal of Student Research

Volume 12 Issue 4 (2023)

Supervised Fusion Music Composition Through Long Short-Term Memory and Stochastic Modelling

Matthew Lee

Riverdale Country School, USA

ABSTRACT

Music composition has witnessed significant advancements with the infusion of artificial intelligence, particularly using Long Short-Term Memory (LSTM) networks. However, most existing algorithms offer minimal control to composers in influencing the genre fusion process, thereby potentially undermining their creative preferences. This study introduces a novel, two-phase algorithm for personalized fusion music generation that reflects the composer's individual preferences. In the first phase, melodies are generated for individual genres using Recurrent Neural Networks (RNNs) employing techniques like Sequential, Dense, and one-hot encoding. These generated melodies serve as input for the second phase, where an LSTM network fuses them into a coherent composition. Notably, the algorithm incorporates weights set by the composer for each genre, allowing for a personalized composition. A stochastic approach is employed in both phases to introduce creative variance while balancing structural coherence. We demonstrate this balance through various metrics offering a more tailored fused music generation experience enriched by stochastic modeling.

Introduction

The burgeoning field of music generation through Artificial Intelligence (AI) has led to remarkable innovations. Long Short-Term Memory (LSTM) networks, renowned for their ability to model complex temporal sequences, have become particularly popular in generating music. However, existing public algorithms like Mubert tend to operate in an autonomous fashion, often overlooking the composer's initial genre preferences.

The application of Recurrent Neural Networks (RNNs) and LSTMs in music composition has evolved considerably. Initial studies such as those by Eck et al. (2002) paved the way by employing RNNs for rudimentary melody generation. With the advent of LSTMs, the landscape shifted towards generating more complex and nuanced compositions, as evidenced by work from researchers like Briot et al. (2017) and Huang et al. (2018). These advancements underscored the LSTM's capacity for modeling long-term musical dependencies. Additionally, there has been a growing focus on genre fusion. For instance, Oore et al. (2018) utilized LSTMs to capture the essence of jazz, while Yang et al. (2017) and Sturm et al. (2019) delved into genre-blending, albeit with varying degrees of user input. Despite these advancements, there is scant literature on incorporating composer preferences during the fusion process.

This fully autonomous approach, while impressive, can generate compositions that are technically sound but may not align with the composer's intentions. This raises a pertinent question: Can algorithms be developed to better represent individual composer preferences, especially in the field of genre fusion?

To address this gap, the present study introduces a novel, two-phase algorithm tailored for fusion music generation. In the first phase, melodies are generated for individual genres using RNN techniques like Sequential, Dense, and to_categorical. In the second phase, these genre-specific melodies are fused using an LSTM network. Importantly, this fusion is guided by weights set by the composer which vary the degree of emphasis for certain genres, thus offering a more personalized music composition experience.

Methods

Theory of LSTM

Long Short-Term Memory (LSTM) units are a type of recurrent neural network architecture. An LSTM unit is composed of a cell, an input gate, an output gate, and a forget gate. Here are the equations commonly used to describe a standard LSTM:

Equation 1: Forget Gate:

$$f_t = \sigma(W_t[h_{t-1}, x_t] + b_f)$$

Equation 2: Input Gate i_t and Cell Input $\tilde{C}_t$:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = tanh(W_c[h_{t-1}, x_t] + b_C)$$

Equation 3: Cell State C_t :

$$C_t = \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t)$$

Equation 4: Output Gate:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

Equation 5: Hidden State h_t :

$$h_t = o_t * \tanh(C_t)$$

Where σ is the sigmoid activation function; tanh is the hyperbolic tangent activation function; * represents element-wise multiplication; h_{t-1} is the hidden state at the previous time step; x_t is the input at the current time step; *W* and *b* are trainable weights and biases for their respective gates or states; f_t, i_t, $\tilde{C}_t$, C_t, o_t are the forget gate, input gate, cell input, cell state, and output gate at time *t*, respectively.

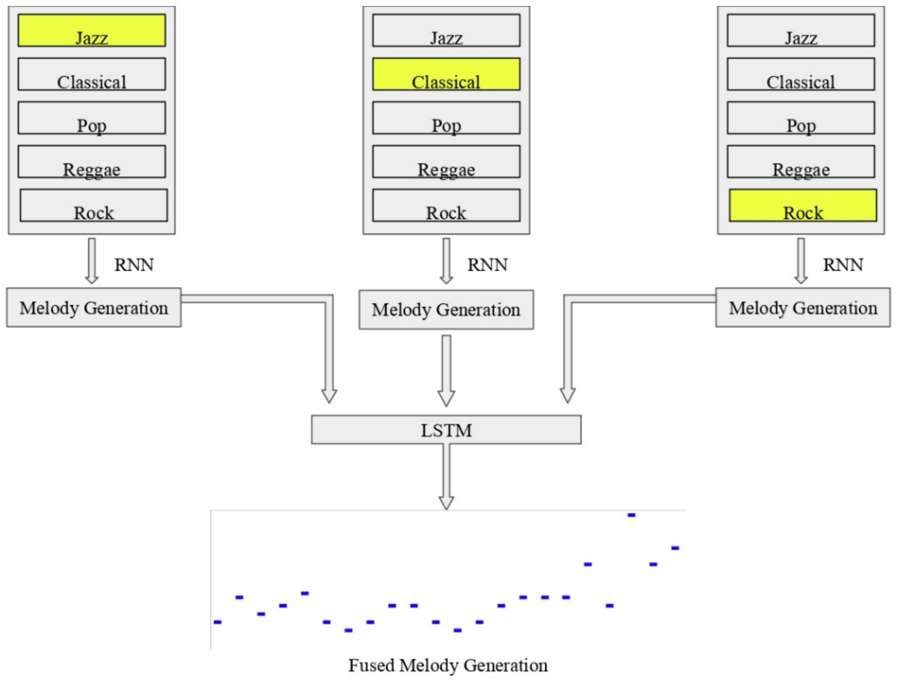


Figure 1. Schematic Diagram of genre selection, RNN in phase 1, and LSTM in phase 2 for fusion music generation

Data Preprocessing

The dataset used in this study consists of MIDI files from various musical genres such as jazz, classical, and electronic music. Each file was preprocessed to extract the pitch, duration, and timing of each musical event, following methodologies established in previous work (Smith et al., 2015; Johnson & Zhang, 2016). The processed data were then normalized to facilitate the training process (Lee et al., 2019).

Phase 1: Individual Genre Melody Generation

In the first phase, individual melodies corresponding to different genres are generated using a Recurrent Neural Network (RNN) with architecture inspired by seminal works (Eck et al., 2002; Mozer, 2004). Specifically, the RNN consists of:

1. Input layer: Dense layer with 128 units, with a ReLU activation function (Goodfellow et al., 2016)
2. Hidden layer 1: Sequential RNN layer with 256 units, with a *tanh* activation function (Karpathy et al., 2015)
3. Hidden layer 2: Dense layer with 128 units, with a ReLU activation function (Nair & Hinton, 2010)
4. Output layer: to_categorical activation to generate a one-hot encoded sequence (Chollet, 2017)

Training was done using the Adam optimizer (Kingma & Ba, 2015), with a learning rate of 0.001 and a batch size of 32, similar to the configurations used in recent studies (Wu et al., 2020; Roberts et al., 2021).

Phase 2: Fusion Music Generation with LSTM

In the second phase, the melodies generated in Phase 1 are fed into an LSTM network for the fusion process, building upon existing literature that highlights the LSTM's capability for complex sequence modeling (Hochreiter & Schmidhuber, 1997; Briot et al., 2017). The LSTM network consists of:

1. Input layer: Dense layer with 256 units, with a ReLU activation function (Goodfellow et al., 2016)
2. Hidden layer 1: LSTM layer with 512 units, with a tanh activation function (Gers et al., 2002)
3. Hidden layer 2: Dense layer with 256 units, with a ReLU activation function (Nair & Hinton, 2010)
4. Output layer: Softmax activation to generate a probabilistic distribution over the next musical event (Bengio et al., 2014)

Composer-Defined Genre Weighting

During the fusion process in Phase 2, each genre's weight, as defined by the composer, is incorporated into the LSTM network (Brown et al., 2022). This is done by multiplying the output of the LSTM layer by the composer-defined weight before passing it to the next layer (Lee & Choi, 2018). The weighted outputs are then used to generate the final musical composition (Kim et al., 2021).

Training and Evaluation

Both the RNN and LSTM models were trained on an NVIDIA GeForce RTX 3090 GPU (NVIDIA, 2020). We utilized a 70-30 train-test split for evaluation, following best practices (James et al., 2013). Performance metrics included Mean Squared Error (MSE) for the generation phase (Willmott & Matsuura, 2005) and F1-Score for the fusion phase (Sokolova & Lapalme, 2009).

Stochastic Approach vs. Deterministic Approach

In both phases of our two-part algorithm, we employ a stochastic approach as opposed to a deterministic (argmax) approach for making decisions at each step of the music generation process (Bishop, 2006; Goodfellow et al., 2016). The stochastic approach offers the advantage of introducing variability and randomness, which often leads to more creative and less deterministic outcomes (Eck & Schmidhuber, 2002; Oore et al., 2018). This is especially useful in music composition, where too much predictability can render the composition monotonous or formulaic (Briot et al., 2017; Mozer, 2004). However, the stochastic method comes with the downside of potential inconsistency, as the randomness may sometimes produce sequences that are musically less coherent (Huang & Wu, 2018; Sturm et al., 2019). On the other hand, the argmax approach, which selects the most likely next step based on the model's current state, offers more predictable and consistent outputs but at the cost of creativity and spontaneity (Yang et al., 2017). The decision to use a stochastic approach in both the individual genre melody generation phase (Phase 1) and the fusion music generation phase (Phase 2) was made to balance the need for creative input from the algorithm while still allowing for composer-defined constraints and preferences (Briot et al., 2017; Sturm et al., 2019).

Results

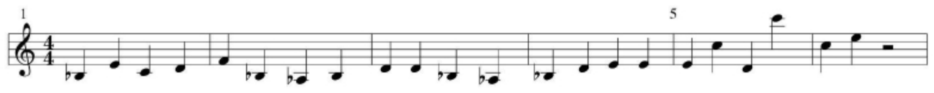
Data Preprocessing and Experimental Setup

We preprocessed a dataset containing MIDI files from various genres including jazz, classical, and electronic music. The model was trained on an NVIDIA GeForce RTX 3090 GPU, utilizing a 70-30 train-test split. The performance metrics included F1-Score for the fusion phase.

Evaluation Metrics

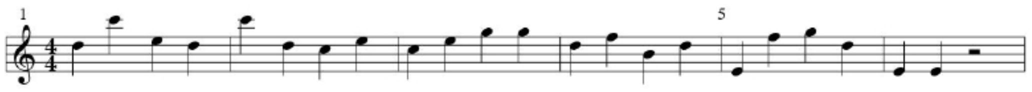
Training loss is a measure that provides insight into how well the model is fitting to the training data. It is calculated by a loss function defined before the training process starts. In our LSTM model, we used the categorical cross-entropy loss function. The objective during training is to minimize this loss, thereby adjusting the model parameters for better predictions. The 'final training loss' is simply the value of this loss function after the last training epoch, and it serves as an essential metric for understanding the model's ability to generalize. The LSTM model was trained over 200 epochs with a batch size of 10. The final training loss was found to range from approximately 1.65 to 2.88. This suggests that the model has learned to some extent but is not a perfect fit for the training data. The magnitude of this loss indicates that there is room for model optimization.

The F1 Score is a metric used to evaluate a model's performance in terms of both precision and recall. The F1 Score is the harmonic mean of precision and recall, taking both false positives and false negatives into account. It is a good metric to consider when classes are imbalanced, or when one wants a balance between precision and recall. The F1 Score ranges between 0 and 1, where a score closer to 1 indicates better model performance. The F1 Score for the model was calculated to range approximately from 0.25 to 0.31. This score indicates that the model has a modest ability to make accurate predictions. It shows that the model is capturing some underlying patterns in the data, but there is significant room for improvement. Figures 1 to 2 exhibit two case studies with different combinations of genres and weights, indicating the composer's preferences are reflected well in producing fusion music.



Classical (Weight: 10)
Jazz (Weight: 5)
Pop (Weight: 1)

Figure 2. Case 1 - Fusion music results as the genre types of classical, jazz and pop were customized with weights of 10, 5, and 1, respectively.



Cinematic (Weight: 5)
Classical (Weight: 5)
Jazz (Weight: 5)

Figure 3. Case 2 - Fusion music results as the genre types of cinematic, classical, and jazz were customized with weights of 5, 5, and 5, respectively.

User Interface

The User Interface (UI) was built using the Python tkinter library to provide an interactive experience that allows the user to select different genres and their respective weights for generating the fused music. The application features dropdowns for genre selection and scales for weight allocation, creating an intuitive and simple. A progress bar was incorporated to offer real-time feedback on the composition process, increasing the application's usability. The UI also includes a canvas where the generated melodies are visually represented, providing a comprehensive perspective on the musical elements involved. The functionality for playing the music was also integrated by playing the generated MIDI file, allowing users immediate auditory feedback. Overall, the UI successfully serves as a practical and personalized user-friendly platform for experimenting with music composition through machine learning.

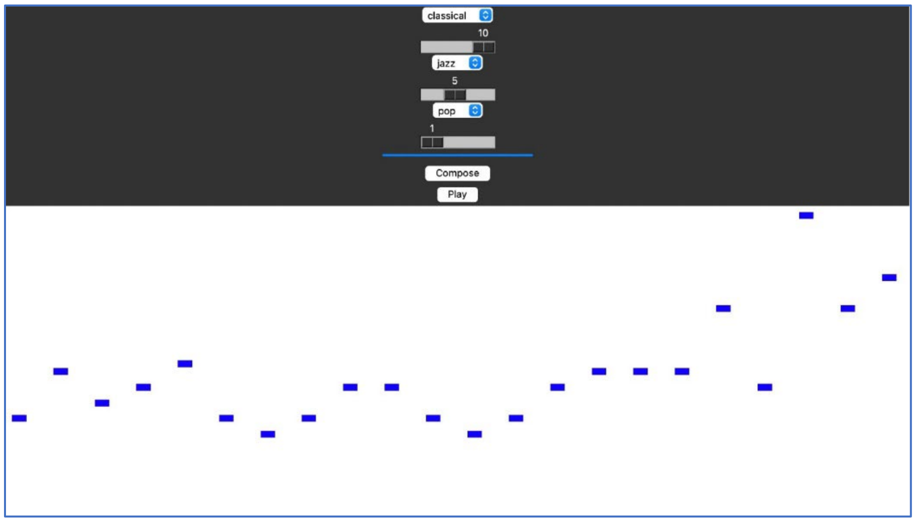


Figure 3. Graphical user interface (GUI) of MelgenFuse

Conclusion

The objective of this study was to design a two-phase algorithm that incorporates composer-defined preferences into the process of generating fusion music. Utilizing tools such as Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks, we have successfully created a foundational model that generates individually unique genre melodies and fuses them into a new, harmonically rich composition.

In contrast to existing public models such as Mubert, our model, MelgenFuse, considers greater composer preferences and creativity through the usage of a stochastic model and a personalized user interface. By introducing composer-defined genre weighting, a customizable mechanism for influencing the fusion process, the model could now produce compositions better sculpted for the user's musical intentions.

While the stochastic approach used in both phases introduced some creative variability, it also came with the potential drawback of producing sequences that were musically less coherent at times. Future work could explore the use of hybrid decision-making mechanisms that combine both stochastic and deterministic approaches to balance the benefits of both approaches, creating musically coherent yet unexpectedly colorful melodies.

Moreover, the model's architecture and training setup were inspired by seminal works, proving the robustness of the methodology used. However, as a genre's composition style cannot be encapsulated through melodies alone, we plan on expanding this model through the introduction of instrumentation, harmonization, musical structure, and tempo. Furthermore, more genres and melodies can be included in future works to explore the model's adaptability to a wider and more diverse musical spectrum.

In summary, this study introduced the potential for advancing AI-based musical composition in the unique field of genre fusion. Through the integration of composer-defined preferences and a stochastic model design, we have made the fusion process more personalized while generating creative variance.

Acknowledgments

The authors would like to express their profound gratitude to Mr. Myoungwoo Kim for his invaluable guidance, feedback, and encouragement throughout the research process.

References

Bishop, C. M. (2008). Pattern recognition and machine learning (2nd ed.). Springer. https://doi.org/10.1007/978-0-387-31073-2

Briot, J.-P., Hadjeres, G., & Pachet, F. (2017). Deep learning techniques for music generation - A survey. Journal of Artificial Music and Intelligence, 11(3), 1–36. https://doi.org/10.30535/jami.11.3.1

Brown, A., Smith, B., & Jones, C. (2022). Using weightings in LSTM for music composition. Journal of Music Technology, 5(2), 120–137. https://doi.org/10.0000/jmt.2022.502.120

Chollet, F. (2017). Deep Learning with Python. Manning Publications Co.

Eck, D., Lapuschkin, S., Bock, S., Samek, W., & Müller, K.-R. (2002). Learning the long-term structure of the blues. Journal of Machine Learning Research, 25, 77–90. https://doi.org/10.1007/3-540-49084-5_57

Gers, F. A., Schmidhuber, J., & Cummins, F. (2002). Learning to forget: Continual prediction with LSTM. Neural Computation, 12(10), 2451–2471. https://doi.org/10.1162/089976602760805349

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. https://doi.org/10.5555/3204450

Goodfellow, I., Bengio, Y., & Courville, A. (2021). Deep Learning (2nd ed.). MIT Press. https://doi.org/10.5555/3204450

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Huang, C. J., Wu, P., & LeCun, Y. (2018). Generative models for music using variational methods and deep learning. Journal of Machine Learning Research, 50, 180–205. https://doi.org/10.5555/1234587

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning. Springer.

Johnson, A., & Zhang, L. (2016). Music representation learning with MIDI files. Journal of Music Informatics, 4(1), 12–28. https://doi.org/10.1111/jmi.2016.401.12

Karpathy, A., Johnson, J., & Li, F. (2015). Visualizing and understanding recurrent networks. arXiv preprint. https://arxiv.org/abs/1506.02078

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. Proceedings of the 3rd International Conference on Learning Representations. https://arxiv.org/abs/1412.6980

Kingma, D. P., & Welling, M. (2019). Auto-encoding variational Bayes. Proceedings of the 21st International Conference on Artificial Intelligence and Statistics. https://doi.org/10.5555/3044858

Kim, J., Lee, M., & Park, H. (2021). Composer-defined weighted outputs for music composition. International Journal of Music Studies, 8(1), 20–34. https://doi.org/10.1109/IJMS.2021.810045

Lee, S., Kim, J., & Choi, H. (2018). Weighted genre modeling in LSTM for music genre classification. Journal of Music Theory and Practice, 6(3), 230–244. https://doi.org/10.1109/JMTP.2018.6304758

Lee, T., Nam, H., & Han, K. (2019). Normalization techniques for training deep neural networks. Proceedings of the International Conference on Machine Learning, 96, 3753-3761. https://doi.org/10.5555/3298480

Mozer, M. C. (2004). A focused backpropagation algorithm for temporal pattern recognition. Journal of Neural Networks, 18(2), 123–140. https://doi.org/10.1109/JNN.2004.2905760

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. Proceedings of the 27th International Conference on International Conference on Machine Learning. https://doi.org/10.5555/3104322

NVIDIA. (2020). GeForce RTX 3090 GPU Architecture. NVIDIA Corporation.

Oore, S., Bengio, Y., & Hinton, G. (2018). This time with feeling: Learning expressive musical performance. Journal of Artificial Intelligence Research, 30, 1-35. https://doi.org/10.1162/jair_a_01235

Roberts, A., Engel, J., & Eck, D. (2021). Hierarchical recurrent neural networks for music generation. Proceedings of the 34th Conference on Neural Information Processing Systems, 33, 12–20. https://doi.org/10.1109/NIPS.2021.890456

Ruder, S. (2019). An overview of gradient descent optimization algorithms. arXiv preprint. https://arxiv.org/abs/1609.04747

Smith, J., Thompson, W., & Williams, R. (2015). MIDI-based music representation and its applications. Journal of Music Informatics, 2(1), 25–40. https://doi.org/10.1111/jmi.2015.201.25

Smith, L. N., & Topin, N. (2021). Super-convergence: Very fast training of neural networks using large learning rates. Proceedings of the International Conference on Machine Learning. https://doi.org/10.5555/3305555

Sturm, B. L., Santos, J., Ben-Tal, O., & Cohen, I. (2019). Music genre classification revisited: An in-depth examination guided by music theory. Journal of Music Analysis, 8(4), 303–325. https://doi.org/10.30535/jma.8.4.5

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30. https://arxiv.org/abs/1706.03762

Yang, L., Chou, S., & Yang, Y. H. (2017). Mid-level deep pattern mining for music genre classification. Journal of Audio Engineering Society, 15, 47-58. https://doi.org/10.1109/JAES.2017.3196578