IJHSR
ijhighschoolresearch.org
RESEARCH ARTICLE

Evaluation of AI-based Music Generation Using an LSTM

Vy N. Nguyen

Haileybury Imperial & Service College, Hertford, Hertfordshire, SG13 7NU, UK; violetnvy08@gmail.com

ABSTRACT: Music generation has been a significant advancement since the introduction of AI, saving users time and effort to produce decent-sounding music. One of the recurring neural network (RNN) models that is capable of producing pleasant music is the Long Short-Term Memory (LSTM). This investigation aims to create an LSTM network that could generate music from a Chopin MIDI dataset and alter its hidden unit size to see which model best generates music and resembles Chopin. Previous studies have used LSTMs in music generation for their capability to predict time series. After rigorous training sessions and architectural choices, we demonstrate that there exists a set of hidden units in the LSTM that can lead to optimum performance. Exploring this balance will provide valuable insights to further advance the field of music generation, particularly in complex genres like classical music. Results will remain preliminary; for further exact hyperparameter optimization, further investigation is needed.

KEYWORDS: Robotics and Intelligent Machines, Machine Learning, Music Generation, LSTM, Hidden Units Dimension.

Introduction

While classical music brings many values and joy to life, composing a classical piece of music requires skill, practice, and years of education. With automatic music generation, a person wouldn't need to spend so much time creating a classical piece.

Music generation using AI has been developed for a long time, and experts in the field have attempted to predict it using various techniques. Recurrent Neural Networks (RNNs) are used for music generation.[1] Unlike the "standard" feed-forward neural network (FNN) architecture of layered loop-free neurons, RNNs build a memory of time series events by enabling data propagation throughout their layers of neurons, lower to higher. This makes it useful for sequential data, data that requires order.[2] Todd (1989),[3] Bharucha and Todd (1989),[4] and many others tried a single-step predictor RNN to predict individual notes by their probability of occurring based on their immediate previous note. However, that method was found to be ineffective. This extends in general to RNNs and FNNs, which also have a limited capability of composing music as they are unable to recall previous notes or the structure of the music prior to the given input.[1] RNNs have received criticism from Mozer (1994),[5] for lacking global coherence, thematic and phrase structure, and rhythmic organization.

This is where the Long Short-Term Memory (LSTM) architecture enters: an extended RNN that possesses gated memory units (input, output, forget gate) which enable them to remember previous information while ignoring unnecessary ones.[6] Numerous studies, such as Jin *et al.*,[7] Kotecha and Young,[8] and Conner *et al.*,[9] employ LSTM networks to generate music due to their effective long-term memory mechanism. This architecture was supported by Chung *et al.*,[10] and Eck and Schmidhuber,[1] that it outperforms RNNs in generating music and is capable of generating quality music.

Before generating music using a computer, an algorithmic composition appeared in Musikalisches Wurfelspiel (attributed to Mozart), which randomly selects segments of music unique to a certain style. All the way to 1957, The Illiac Suite for the String Quartet was the first piece to be composed using artificial intelligence. Its algorithm randomly generates numbers that are associated with features such as rhythm or pitch, but the randomness was restricted by the rules of music theory.[11]

Soon, in the 1980s, composer and professor David Cope saw potential with computer composition and based his work on recombancy, where we take elements of previous works and combine them to make new works. Musical features are encoded into databases, and the extracted patterns are then reconstructed into new logical pieces. This is a foundation for many modern AI models.[12]

Fast forward to today, music generation is very much developed, especially in its creative process. Iamus, developed by Melomics Records, is the first computer to compose classical music in its own style instead of recombining examples. It uses a computer cluster, an evolutionary algorithm that mimics how humans compose music biologically.[13] Other innovations like Magenta by Google Brain use the machine learning model MusicVAE to tackle the problem with long-sequence structure in music generation by encoding compact vectors of notes and then decoding them by embedding complexity to the subsequences. In this way, it produces significantly better results than a flat baseline RNN VAE model. Even with this advanced model developed by Google, the LSTMs are the core of this model. They are used in the MusicVAE's encoder, conductor, and decoder in an RNN-like structure.[14]

To generate quality music is to generate pleasant-sounding music to the ear that could be evaluated with musical features. There lacks a systematic study on how to optimize music generation using LSTMs, specifically for model complexities. This article aims to explore how the number of hidden units affects LSTM's music generation performance. Nevertheless, devel-

DOI: 10.36838/IJHSR89.271

oping the area of classical music generation will bring many benefits that music brings to us.

As music is subjective, we cannot rely on parameter tuning or optimization tools, as quantitative metrics cannot capture creativity and novelty. Hence, to evaluate which number of hidden units yields the best performance, a survey will be sent to 17 musicians, students, and professionals to evaluate the excerpts produced in many categories (eg Musical features, Overall sentiment, Chopin Resemblance).

The rest of the introduction will give a brief overview of LSTMs, AI in music generation, and a brief overview of the hyperparameters of the model.

Recurrent Neural Networks (RNNs) and LSTMs:

Although RNNs can temporarily store data, they cannot retain data for more than 5-10 steps.[2,15] This is due to the vanishing gradient problem identified by Bengio *et al.* (1994),[16] and Hochreiter (1991).[17]

When training algorithms, there are parameters to optimize the model's accuracy when predicting. Those are the weights (denoted as W) and biases (denoted as b). In an unfolded RNN, backpropagation through time can be used to train the weights. It does this by propagating from the output layer to the initial layer and calculates each weight's contribution to the error function. $\theta_u(\tau)$ is the error signal and is obtained by calculating the derivative of

$$\frac{\delta E(\tau)}{\delta z_u(\tau)}$$

where E is the error function, $z_u(\tau)$ is the function of total input to unit u at time τ, and τ are the discrete time steps between t' (the time when the network starts) and t (the time when the network ends).[2,16,18] When we apply the chain rule to that function, the error gradient is:

$$\frac{\delta E(\tau)}{\delta z_u(\tau)} = \frac{\delta E(\tau)}{\delta y_u(\tau)} \cdot \frac{\delta y_u(\tau)}{\delta z_u(\tau)}$$

where $y_u(\tau)$ is the output of the unit u at time τ, with f_u being a differentiable activation function, the equation is expressed as follows:

$$y_u(\tau) = f_u\big(z_u(\tau)\big)$$

With that, the error signal function can be rewritten as:

$$\frac{\delta E(\tau)}{\delta z_u(\tau)} = \frac{\delta E(\tau)}{\delta y_u(\tau)} \cdot f'_u(\tau)$$

Here, the derivative of the activation function, $f_u(\tau)$, is a scaling factor of the gradient. As this scaling factor drops below 1, the gradient of the error function vanishes exponentially as the RNN processes through the network, causing error loss. On the other hand, if the activation slope goes above 1, it blows up the network.

The LSTM model was introduced by Hochreiter and Schmidhuber (1997),[19] as a solution to the vanishing gradient problem, capable of remembering long-term information better than typical RNNs. This is achieved by using constant error carousels (CECs), which ensure that the gradient flows unchanged throughout the network by keeping the error signal closer to 1.[2] Like the RNN family, LSTMs have hidden states that are passed from one step to another in a recurrent structure.[20] One unique characteristic of the LSTM is its architecture: it effectively retains necessary information while eliminating unnecessary information.

LSTMs have high capabilities for handling sequential data, where the length is not fixed and the order and timing matter. Hence, they are commonly used for time series forecasting,[20,21] including stock prediction, weather forecasting, speech recognition, and many more. Recently, they have been widely used for music generation, which is also a time series (sequential data).

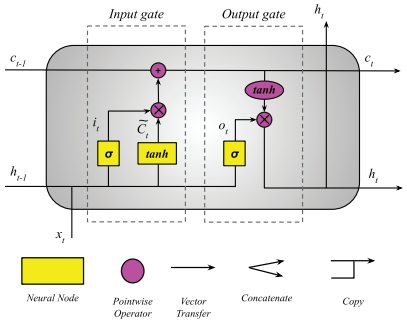

Figure 1: Diagram of an LSTM cell with input-output gate.

Figure 1 shows the architecture of the original LSTM implementing the sigmoidal and tanh functions as discussed above. The equations are as follows.

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

$$\tilde{C}_t = tanh\left(W_{\tilde{C}} \cdot \big[h_{t-1}, x_t\big] + b_{\tilde{C}}\right)$$

$$C_t = C_{t-1} + i_t \cdot \tilde{C}_t$$

$$o_t = \sigma\Big(W_0 \cdot \big[h_{t-1}, x_t\big] + b_o\Big)$$

$$h_t = o_t \cdot tanh\,(c_t)$$

Its original architecture involves an input and output gate, where the input gate decides how much new information can be stored in the new cell state, and the output decides what information is output.[6] The sigmoidal function would produce values from 0 to 1, which decides how much information is allowed in/out of the cell; it will eventually multiply with vectors when approaching an operation. The tanh function would squash each vector into values from -1 to 1, which eventually decides what will be output.

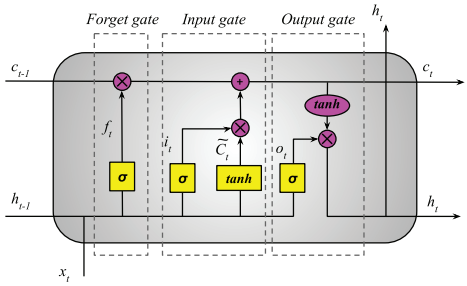

Figure 2: Diagram of an LSTM cell with a forget gate.

Figure 2 shows the cell architecture with a forget gate using sigmoidal and tanh functions. Its equations are as follows.

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right)$$

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

$$\tilde{C}_t = tanh\left(W_{\tilde{C}} \cdot \left[h_{t-1}, x_t\right] + b_{\tilde{C}}\right)$$

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

$$o_t = \sigma\left(W_0 \cdot \left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t \cdot tanh\left(c_t\right)$$

The architecture with the added forget gate that is widely used is the one employed by this experiment. Gers, Schmidhuber, and Cummins (2000),[15] have added the forget gate to eliminate unnecessary information from the cell state.

The sigmoidal function of the forget gate inputs a number between 1 and 0, 1 indicating full information retention and 0 indicating no information retention.[6] Adding the forget gate will prevent the cell from growing indefinitely, which will prevent the network from breaking down.[15]

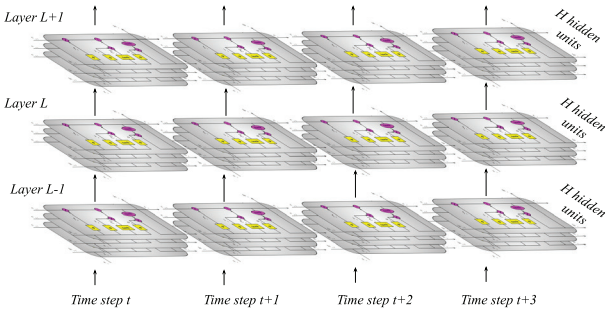

Figure 3: Diagram of a stacked LSTM structure.

The simplest way to increase depth and complexity is to stack LSTMs. Figure 3 shows the structure of the stacking LSTMs. Its equations are as follows:

$$f_t^L = \sigma\left(W_f^L \cdot \left[h_{t-1}^L, x_t^L\right] + b_f^L\right)$$

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

$$\tilde{C}_t^L = tanh\left(W_{\tilde{C}}^L \cdot \left[h_{t-1}^L, x_t^L\right] + b_{\tilde{C}}^L\right)$$

$$C_t^L = f_t^L \cdot C_{t-1}^L + i_t^L \cdot \tilde{C}_t^L$$

$$o_t^L = \sigma\left(W_0^L \cdot \left[h_{t-1}^L, x_t^L\right] + b_o^L\right)$$

$$h_t^L = o_t^L \cdot tanh\left(c_t^L\right)$$

For its simple and effective structure, this experiment will employ stacked LSTMs to generate music, as it is a complex process.

AI in Music Generation:

Generating music is incredibly complex and difficult, as there are many rules, creativity, and understanding of musical styles involved, which is hard for a human to do, let alone AI. Even with advanced architectures, Briot *et al.*,[22] addressed many issues with music generation architectures regarding control, structure, creativity, and interactivity.

There are general concerns and debates regarding the creativity and authorship of AI-generated music. These issues are all addressed by Briot *et al.*[22] From melody to tempo, the person generating music has significantly less control over the final output compared to a human composer. Theoretically, they should have no ownership as they did not engage in a deliberate creative process. Similar to creativity, music generation architectures have to find a point between "junk" and "plagiarism" if they learn from existing data.

The Role of Hyperparameters:

The LSTM has many hyperparameters that could be tuned to influence how well it performs. Hyperparameters are configurations made before the learning process that control how the model learns. These hyperparameters include, but are not limited to: Number of hidden layers, number of hidden units, dropout rates, learning rate, momentum, and sequence length. Previous works like Greff *et al.*,[23] have investigated the importance of hidden unit sizes in an LSTM polyphonic music modelling structure. Their results have suggested that larger hidden unit sizes perform better and are very important for the performance of tasks due to the increased complexity and capability of learning. This study specifically focuses on the number of hidden units as a hyperparameter to optimize music generation. Although larger networks achieve better results, they eventually yield 'diminishing returns' and have a longer training time.[23] There is a trade-off between performance and computational cost; hence, if this phenomenon happens in this experiment, the balance of performance and computational cost is the key to this study

Methods

Dataset and Preprocessing:

The dataset comprised MIDI files of Chopin that were taken from The Classical Piano Midi Page.[24] There are two types of music representation: MIDI and audio files. MIDI files are digital files that instruct how music should sound. Table 1 shows the pieces that comprise the dataset. In here, there are 57,894 notes.

Table 1: Pieces in the Midi Dataset.

Musical Form	Opus number	Count
Preludes	28	24
Mazurkas	7	2
Waltz	18	1
Scherzo	31	2
Etude	10	3
Ballade	23	1
Etude	25	6
Nocturnes	27	2
Mazurkas	33	2
Piano Sonata	35	2
Polonaise	53	1
Impromptu	66	1

DOI: 10.36838/IJHSR89.271

Model:

The architecture implements a many-to-one model, where input sequences composed of individual notes and chords are parsed and concatenated into an array. The data is segmented into overlapping samples with a fixed look-back window size of 100 notes, generating 57,794 unique samples with an output dimension of 3,372. An input consists of 100 notes, where the LSTM will predict the next note following the sequence. Its feature dimension is a scalar value between 0 and 1 ([0,1]). The training was done on the entire dataset.

We implemented a three-layer, unidirectional LSTM to evaluate the effects of the LSTM's unit size on the quality of the composed music. The experiment takes place in Google Colab using the Python programming language. The module for LSTM cells is Tensorflow Keras, and for music is Music21.

Figure 4 is a visualization of the LSTM model that will remain constant throughout the experiment, where H is the only hyperparameter that changes.

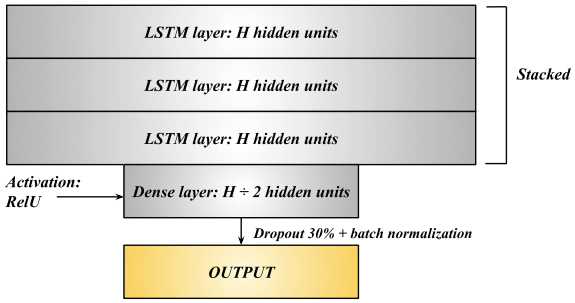

Figure 4: The LSTM model used to create music in this experiment.

Model Training:

There are also some training hyperparameters. Epochs will be set to 100, sequence length is set to 100, loss function is categorical cross-entropy, the optimizer is rmsprop, the learning rate is 0.001, and batch size will be set to 128.

Experimentation:

We will run a lab experiment of a music generation LSTM model and tune the model's number of hidden units (H) in increments of 128 to have a wider range of model sizes. The hyperparameters of H will be 256, 384, 512, 640, and 768. The model state for the epoch with the lowest training loss is stored and used at inference time. We will randomly initiate a sequence of 100 notes, then have the model output the next 400 notes, then send them to a panel of judges to evaluate. With that, Table 2 below will show the details about each model's sizes.

Table 2: Parameters and sizes of each LSTM model.

Hidden Units	LSTM Layers	Trainable Parameters	Total Parameters
256	768	1,783,212	1,782,956
384	1152	3,680,364	3,680,748
512	1536	6,249,516	6,250,028
640	1920	9,490,412	9,491,052
768	2304	13,403,052	13,403,820

We predict that either hyperparameter 512 or 768 will perform best in this experiment.

We hypothesize that **there will be an optimal point in the size of hidden units that would produce the best music, as it balances between incoherent and a lack of originality. This optimal point is in the middle or at the end of the list of H.**

In this experiment, hidden units of 512 could perform best, as the higher the number of hidden units, the higher the chance of similarity, which could increase plagiarism, or it can lack creativity, novelty, and quirks that a human can have. Anything below 512 could have too much loss, hence lacking musical elements, and might sound incoherent. 512 hidden units can be an optimal spot to generate music as it balances between "Chopin music" and "junk" the most, as proposed by Briot *et al.* (2020).[22]

If it is not 512 hidden units outperforming, we predict that 768 hidden units could do the best, as it has more parameters, more complexity, which means that it has a higher capability to learn Chopin's music than the other hyperparameters. If this is the case, it can have plateau-like results similar to those of Greff *et al.* (2017).[23]

It is to note that the prediction of the hyperparameters of 512 and 768 hidden units performing best in this experiment will be preliminary. Only further research can determine what exactly is the best hidden unit size hyperparameter, as that is how hyperparameter optimization works.

The survey was sent out to 17 individuals, 13 of whom are students, teachers, or performers of classical music. The other 4 are classical music enthusiasts or teachers/students in another music genre.

To remove bias and make the concept easier to understand, we have told the takers that we are testing AI "Models" instead of parameters. We have associated the 5 different hidden unit values with numbers 1 to 5 in a randomized fashion. Table 3 shows the order of models.

Table 3: Table of models and their hidden unit size.

Model	Hidden Units
Model 1	640
Model 2	384
Model 3	256
Model 4	512
Model 5	768

We also give the takers multiple excerpts to see the variety of music the "model" could produce. Each model generated 3 excerpts, which were screened for similarity. If two excerpts were deemed to be too similar, then an extra one was replaced by a newly generated excerpt until all 3 pieces are sufficiently different. In the end, each survey taker received 15 excerpts, 3 for each model.

After the subjects listen to each batch of excerpts generated from the model, the survey asks them to rate from 1-10 for each present in the piece: melody, rhythm, character/emotion, harmony, musical texture, phrasing/structure, coherence, novelty/creativity. 1 being "a mess", "unlistenable", and 10 being "very distinct". We also ask the survey takers about their overall sentiment of the music produced by each model from 1-10, if they think this model produces classical music (yes/no), and rank from 1-10 how much the music produced by the model resembles Chopin. In the end, they were also asked to rank

each model from 1-5, with 1 being the best and 5 being the worst.

Results and Discussion

After training all models, Figure 5 shows the training losses for each model variant based on the choice of hidden dimension, and Table 4 shows the minimum loss of each model.

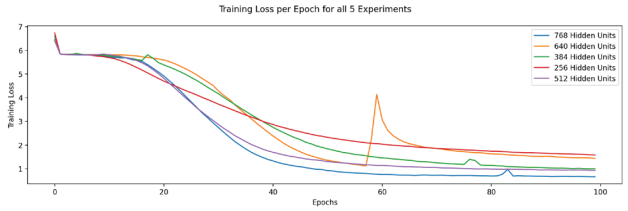

Figure 5: Training Loss of all 5 models per epoch.

Table 4: Hidden Units and Corresponding Minimum Loss.

Hidden Units	Minimum Loss
256	1.587
384	0.999
512	0.937
640	1.116
768	0.663

Below is Table 5 showing the average score for individual musical features, Table 6 showing the overall scores for each hyperparameter, and Table 7 showing the overall rank test takers have given.

Table 5: Average score for each musical feature across different numbers of hidden units.

	Number of Hidden Units				
	256	**384**	**512**	**640**	**768**
Melody	3.824	3.294	**6.529**	3.941	5.294
Rhythm	4.294	4.471	**5.765**	4.882	5.000
Character/ Emotion	2.706	3.118	**4.353**	3.471	3.706
Harmony	3.647	3.353	**6.235**	5.294	5.176
Musical Texture	3.294	3.529	**5.000**	4.118	4.235
Phrasing/ structure	3.176	3.118	**4.588**	3.647	4.353
Coherence	3.412	3.588	**4.941**	3.882	4.412
Novelty/Creativity	3.882	3.882	**5.353**	4.765	4.647

Table 6: Overall scores for each hyperparameter of hidden units.

	Number of Hidden Units				
	256	**384**	**512**	**640**	**768**
Average Score of Features	3.529	3.544	**5.346**	4.25	4.603
Average Overall Sentiment	3.412	3.412	**5.529**	4.294	4.882
% Classifying output as Classical Music	41.2%	35.3%	**64.7%**	52.9%	**64.7%**
Average score of Chopin resemblance	2.882	3.118	**4.941**	3.824	4.647

Table 7: Average Ranking given by Test Takers for each Hyperparameter.

Hidden Units	Rank
256	3.41
384	3.94
512	1.94
640	3.59
768	2.12

Below is Table 8 showing the standard deviation for each measurement.

Table 8: Standard deviation of each quantitative metric.

	Number of Hidden Units				
	256	**384**	**512**	**640**	**768**
Melody	1.912	2.144	1.700	2.436	2.144
Rhythm	2.339	2.401	1.855	2.395	2.151
Character/Emotion	1.572	1.996	2.178	2.452	1.863
Harmony	1.869	1.801	1.602	2.339	2.157
Musical Texture	1.829	1.875	2.121	2.315	2.306
Phrasing/Structure	1.976	1.996	2.181	2.473	2.149
Coherence	1.970	1.805	2.015	2.288	2.320
Novelty/Creativity	2.759	2.643	1.967	2.705	2.499
Overall Sentiment	1.970	1.502	1.700	2.144	2.147
Classified as Classical	0.507	0.493	0.493	0.514	0.493
Chopin Resemblance	1.691	1.965	1.713	2.069	2.234

For better visualization, Figures 6 and 7 visually display the results based on Tables 5 and 6. They show the performance of each model across various evaluation metrics. Specifically, Figure 6 shows the result assessed through individual musical features, while Figure 7 shows the result according to broader metrics such as total sentiment, Chopin resemblance, average feature scores, and classical music resemblance.

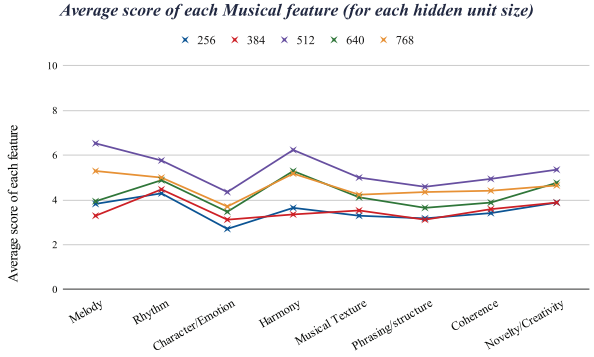

Figure 6: Average scores of each musical feature (Based on Table 5).

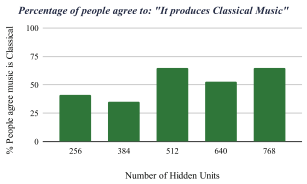

Figure 7a: Percentage of people agreeing that the models produce classical music.

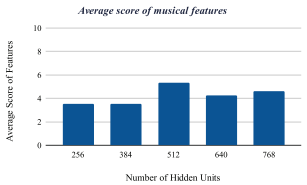

Figure 7c: Average score of musical features for each hidden unit size.

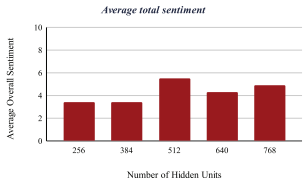

Figure 7b: Average total sentiment of each hidden unit size.

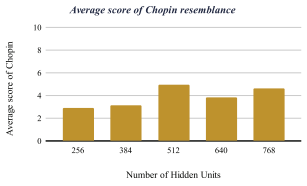

Figure 7d: Average score of Chopin Resemblance for each hidden unit size.

Figure 7: Scores for each hidden unit size (based on Table 6).

DOI: 10.36838/IJHSR89.271

Analysis:

As seen from Tables 5, 6, and 7, the model with 512 hidden units consistently outperformed the others, with 768 being a close second. Both models are equally assessed to be producing classical music by 64.7% of survey takers. However, the scores show a consistent trend (from best to worst: 512, 768, 640, 384, 256), and the ranking (Table 7) of parameters 512 and 768 slightly differed for the bottom 3 models, which could be due to bias or personal preference/priority.

It is to note that the 3 lowest ranked models only differed by 0.53, which is trivial. However, the gap between 256 hidden units (3rd-ranked model) and 768 hidden units (2nd-ranked model) is quite large (1.29). The rankings had a slight difference between 512 and 768 hidden units, with some saying that 768 better resembled Chopin and had more harmony than the rest.

The scores that respondents give for the models range between 3 and 6, which means that all models produced bad to mediocre music. Many of the comments given were related to the music's melody's lack of creativity and distinctiveness compared to Chopin's 'beautiful lyricism' and 'visceral romanticism'. Furthermore, rhythm, structure, and phrasing were also described as 'bland', 'monotonous', and 'singular', which hints that this model should be further improved. Some excerpts do possess structure and some repetition of the main theme, but overall were very repetitive and indistinctive.

The melody changed as the hidden units differed. Some of the comments mentioned that only Models 4 (512 hidden units) and 5 (768 hidden units) had some hints of harmony and coherence, but the rest are incoherent.

There were also major concerns with a lack of originality, as there was a lot of similarity, especially with higher hidden units. After having a further investigation (see Section 3.3), many excerpts produced were similar to Chopin's work, such as Etude Op. 10 No. 12, Op. 24 No. 4 & 11, Fantasie Impromptu, Ballade No. 1, and possibly many more. The music produced by these models lacked novelty or originality from the training dataset.

Observations and discussions:

Although it is clear that the model with 512 hidden units outperformed in this experiment, it was not definitive, as there is an element of randomness to each generation. Some excerpts were musically coherent and ear-pleasing, while others were considerably weaker, even when they came from the same model. For example, the model with 640 hidden units did not perform well in the experiment, but its third excerpt was rated favorably by an independent person with orchestra experience who was not a test taker. This highlights a wide range of quality that can be produced by the same model. Still, the amount of decent-sounding music appeared more in models with a hidden unit dimension greater than or equal to 512.

Further Investigations:

The results supported our hypothesis that the model with 512 hidden units is the best-performing hyperparameter in this survey.

Further experiments were conducted after the survey with hyperparameters of 864 hidden units and 1024 hidden units to ensure a downward trend continued with the increased hidden dimension size. These larger models did not show a distinguishable difference from those with 512 or 768 hidden units through listening.

An important qualitative observation was that overall quality diminished after approximately nine bars for larger models. The first nine bars would sound ear-pleasing and musically coherent, but after that, the music is incoherent. The model with 512 hidden units also exhibits the same behaviour for some excerpts, whereas smaller models would start and stay incoherent throughout. We hypothesize that the causation of the phenomenon could be due to a lack of data points, insufficient sequence length, or the limitations of a flat LSTMs baseline model (compared to hierarchical models),[14] to accurately predict the next note. Although some excerpts stay consistent throughout, this is rare.

Limitations & Improvements:

Many subjects were critical of the structure/phrasing, dynamics, and melody aspects of the generated music. Improvement in these areas can be achieved in a variety of ways.

To address the problem of disjointed melody, more data can always be put in order to improve the model. Currently, the model only has around 57,894 notes, which could be too small a data set for the LSTM to effectively predict notes. The model will benefit a lot from a larger dataset, even with classical music, not specifically from Chopin.

Furthermore, implementing rule-based models to ensure that it follows the fundamentals of music theory could be beneficial to prevent it from sounding like a random aggregation of notes. Adding new features, such as a key, could also significantly enhance the model's ability to determine the musical key, providing information about its tonic note and distinguishing between major and minor scales. This will open doors for better chord progression and would prevent clashing notes, hence simultaneously improving harmony.

An improvement regarding structure or phrasing could be to implement rules (Cadences, rest), sequence duration, or additional features for the model to know which phase of the melody it could be in. In that way, it could have a more distinctive start, end, and possibly climax. Implementing an encoding-decoding system like MusicVAE,[14] could also be viable.

Although dynamics/velocity were significant in music, it was not included in the survey because no dynamics were found in any of the excerpts. Having no dynamics was a significant problem as classical music has hidden melodies, voices, and melody chords that require some notes to be prominent while keeping some hidden in the background. Dynamics can help exaggerate climaxes and melancholic moments, and it is arguably one of the most "human" aspects of music. Adding additional features for the model to identify climaxes, phrasing, sequences, hidden melodies, or even arpeggios could help make the music sound more human.

Conclusion

In this paper, we have experimented with various hidden unit sizes of the LSTM model to optimize music generation. The architecture of the LSTM model is a 3-layer LSTM stack. To measure music quality, a survey is sent out to 17 musically educated individuals to evaluate each model based on the provided excerpts.

The results showed that 512 hidden units outperformed all others in all aspects of music asked in the survey. It is definitive that the lower the hidden units, the less it can create quality music. There were also general comments on the quality of music produced, which this paper has addressed and suggested improvements regarding structure, melody, harmony, and dynamics. Our experiment concluded that there was an optimum point for the number of hidden units, although the results remain preliminary. This work can give general insights into optimizing music composition models by examining the effect of increasing the number of hidden units.

Due to the limited scope of the study and the small sample size, more research is needed to better generalize the effect of hidden units on music quality, as well as to find the **optimal number of hidden units** for music generation, given a specific architecture. Furthermore, we consider this a pilot study, only for foundational evidence instead of a definitive conclusion. Nevertheless, the experiment has shown the LSTM architecture's potential to generate decent-sounding music, as well as providing a foundation for future research regarding optimum hyperparameters in music generation models.

Acknowledgments

We first thank Dr. Eric Sakk and Ahmed Shaaban for guiding us through this research paper. This paper would not be possible without their support and guidance. We also sincerely thank those who dedicated their time and participated in the survey. Our deepest appreciation goes to all others who supported this research.

References

1. Eck, D.; Schmidhuber, J. A first look at music composition using LSTM recurrent neural networks. *Istituto Dalle Molle Di Studi Sull'Intelligenza Artificiale* **2002**, *103*, 48–56.
2. Staudemeyer, R. C.; Morris, E. R. Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks. 2019; http://arxiv.org/abs/1909.09586, arXiv:1909.09586 [cs].
3. Todd, P. M. A Connectionist Approach to Algorithmic Composition. *Computer Music Journal* **1989**, *13*, 27–43, Publisher: The MIT Press.
4. Bharucha, J. J.; Todd, P. M. Modeling the Perception of Tonal Structure with Neural Nets. *Computer Music Journal* **1989**, *13*, 44–53, Publisher: The MIT Press.
5. MOZER, M. C. Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multi-scale Processing. *Connection Science* **1994**, *6*, 247–280, Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/09540099408915726.
6. Yu, Y.; Si, X.; Hu, C.; Zhang, J. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation* **2019**, *31*, 1235–1270.
7. Jin, C.; Tie, Y.; Bai, Y.; Lv, X.; Liu, S. A Style-Specific Music Composition Neural Network. *Neural Processing Letters* **2020**, *52*, 1893–1912, Publisher: Springer Science and Business Media LLC.
8. Kotecha, N.; Young, P. Generating Music using an LSTM Network. 2018; http://arxiv.org/abs/ 1804.07300, arXiv:1804.07300 [cs].
9. Conner, M.; Gral, L.; Adams, K.; Hunger, D.; Strelow, R.; Neuwirth, A. Music Generation Using an LSTM. 2022; http://arxiv.org/abs/2203.12105, arXiv:2203.12105 [cs].
10. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. 2014; http://arxiv.org/abs/1412.3555, arXiv:1412.3555 [cs].
11. Hiller, L.; Isaacson, L. M. L. M. *Experimental music; composition with an electronic computer*; New York, McGraw-Hill, 1959.
12. Cope, D. Recombinant music: using the computer to explore musical style. *Computer* **1991**, *24*, 22–28.
13. Quintana, C. S.; Arcas, F. M.; Molina, D. A.; Rodríguez, J. D. F.; Vico, F. J. Melomics: A Case-Study of AI in Spain. *AI Magazine* **2013**, *34*, 99–103.
14. Roberts, A.; Engel, J.; Raffel, C.; Hawthorne, C.; Eck, D. A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music. 2019; http://arxiv.org/abs/1803.05428, arXiv:1803.05428 [cs].
15. Gers, F. A.; Schmidhuber, J.; Cummins, F. Learning to Forget: Continual Prediction with LSTM. *Neural Computation* **2000**, *12*, 2451–2471.
16. Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* **1994**, *5*, 157–166.
17. Hochreiter, S. Untersuchungen zu dynamischen neuronalen Netzen. *Diploma, Technische Universität München* **1991**, *91*, 31.
18. Pascanu, R.; Mikolov, T.; Bengio, Y. On the difficulty of training recurrent neural networks. Proceedings of the 30th International Conference on Machine Learning. 2013; pp 1310–1318, ISSN: 1938-7228.
19. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural computation* **1997**, *9*, 1735–1780.
20. Mojtahedi, F. F.; Yousefpour, N.; Chow, S. H.; Cassidy, M. Deep Learning for Time Series Forecasting: Review and Applications in Geotechnics and Geosciences. *Archives of Computational Methods in Engineering* **2025**, *32*, 3415–3445.
21. Wen, X.; Li, W. Time Series Prediction Based on LSTM-Attention-LSTM Model. *IEEE Access* **2023**, *11*, 48322–48331.
22. Briot, J.-P.; Pachet, F. Deep learning for music generation: challenges and directions. *Neural Computing and Applications* **2020**, *32*, 981–993.
23. Greff, K.; Srivastava, R. K.; Koutník, J.; Steunebrink, B. R.; Schmidhuber, J. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems* **2017**, *28*, 2222–2232, arXiv:1503.04069 [cs].
24. Piano midi.de Chopin MIDI Files and Audio Formats. http://piano-midi.de/chopin.htm, n.d.; Accessed August 23, 2025.

Authors

Vy Nguyen grew up as a pianist, but recently found an interest in STEM, particularly mathematics and data science. She studies at Haileybury and Imperial Service College in the United Kingdom and has hopes of pursuing STEM while keeping music as a hobby.

DOI: 10.36838/IJHSR89.271