Comparison of Deep Learning Models for Melody Generation.

Tatsunori Hirai *

Abstract

Generative deep learning models have been proposed in various media fields, and their generative quality has been improving yearly. The quality of generative models in music generation has been improved, such as “the number of examples of sound generation that deviate from music theory has been reduced” and “melodies that take context into account can be generated.” However, it is difficult to evaluate the “quality of the generated results” in the case of music generation. In this study, by implementing various generative deep learning models, we do not evaluate the quality of the generated results but rather make an objective comparison of each model.

Key words: Generative deep learning model; music generation; deep neural network; variational autoencoder; generative adversarial network

Introduction

With the development of generative deep learning models such as variational autoencoder (VAE) and generative adversarial network (GAN), generation of new contents has become possible, and the quality of generation is improving daily. In the field of content generation, great achievements have been made in image generation and natural language text generation, as well as music generation. The quality of generation has been improving. For music generation, the quality of the output from so-called automatic composition techniques seems to be improving yearly.

One of the major challenges in automatic composition research has long been the difficulty in evaluating generation results. Although research on automatic music composition techniques has been conducted since the early days of computers, it remains a difficult problem to accurately compare and evaluate the generated results. As a result, it is unclear how much progress has been made in music generation technology to date. In this context, taking melody generation technology as an example, results generated by recent state-of-the-art technology tend to be perceived as “good melodies” in terms of audibility. Experimental results comparing melodies generated by the latest generation models with those by real composers show that both generated results are comparable to those generated by humans. Although it can be argued that melody generation technology has evolved in recent years based on such results, it is not necessarily appropriate in the field of music appreciation or music production to conclude that a generated melody is better than a human-made melody solely based on subjective evaluation results. For example, compared to a person with no musical knowledge, an old rule-based automatic composition algorithm could sufficiently generate “good melodies.” In other words, even without recent music generation technology, it is thought that melodies comparable to those of humans could be created, and claiming that it is a good model based on this is not necessarily a sufficient argument. Certainly, recent improvements in generation quality have not been limited to such a...
level. According to the evaluation results of DeepBach\(^1\), one of the generative models, a listening test was conducted on 1,272 subjects, and a large percentage of them answered that the results generated by the system were composed by Bach, suggesting that the distinction between humans and machines is reaching a point where it is difficult to distinguish between the two. On the other hand, there was a large difference in the evaluation results depending on the musical experience of the subjects. The percentage of subjects with vast musical experience who could distinguish between the generated results and Bach's music was high and was lower in subjects with less musical experience. In subjective evaluation of listening experiments, the influence of the differences in the collected subjects on the evaluation results is not negligible, and it is an extremely difficult task to fairly evaluate whether the model can generate “better melodies” from such a viewpoint.

To make matters even more difficult, several recent generative deep learning models have been designed in such a way that changing only one part of the model architecture or tuning only one parameter can significantly change the generated results. In the extreme case, if a researcher arbitrarily chooses the generated result to be used in the experiment from several melodies generated, the evaluation result can easily change. Even if the samples used in the experiment were perfectly randomly selected, we cannot deny the possibility that the evaluation results could have been improved by the tuning process because tuning of hyperparameters leading to the results generated is often performed manually. For example, the evaluation results of models such as BERT\(^2\) and GPT-3, which have produced remarkable results in the field of natural language processing (NLP), show that it is difficult to make a general judgment regarding which model is superior\(^3\). This is because each model has its strengths and weaknesses depending on the task, and although it is possible to evaluate which model is better by focusing on a specific task, the results can easily change depending on factors such as the type of training data and number of training parameters. Because many tasks in the field of NLP have metrics and standards for measuring accuracy, it is possible to fairly judge whether a model is good or bad, but even so, it is difficult to say definitively which is better. The fact that it is even more difficult to evaluate music is evident from the fact that people’s evaluation of music will never be the same.

Because it is unclear whether the improvement of a model has led to the generation of better melodies, research on music generation deep learning models tends to follow the trend of applying techniques that have already been successful in NLP and image generation to music generation based on “good results in other fields.”

There are two directions of evolution in music generation technology. One is the pursuit of techniques to better train a model, and the other is the pursuit of techniques to simply generate “good melodies.” The former is not limited to music information processing but relates to advances in machine learning technology. For example, gated recurrent neural network (RNN), such as long short-term memory (LSTM), learn better than simple RNN because they are less prone to gradient explosion and gradient vanishing problems and they can consider distant contexts by introducing an attention mechanism. These are elements that advance learning technology, but how much they affect the results in music generation is unknown, and whether they contribute to improving music generation and “good melodies” is another story. Although there is no doubt that in

---

developing a generative model, it is better to learn effectively and to be able to take context into account than not, determining the importance of each module is difficult.

At first, when we attempt to implement a generative model to automatically generate a melody, we tend to end up with an unlistenable sequence of sounds in many cases. With more advanced and modern technologies, we can imagine that the percentage of the output that is unbearable to listen to is decreasing, but adjustment of models to improve the percentage often proceeds subjectively based on developers. RNN-based prediction methods, for example, are sometimes evaluated by an objective measure of how accurately they can predict melody that follows the melody of a piece in a dataset, but a model that can correctly predict the melody of an existing piece is not necessarily a good melody generator. In addition, even if the output is audibly decent, there are cases where it is due to overfitting and the generator is generating a part of the training data as it is, and even if it is not overfitting, we cannot say much more than that the generator generated a reasonable melody that can realize the task correctly. It is not surprising that melody sounds “good” if it is generated on the basis of the melody rules acquired from training data, but in this case, it is questionable whether it is a good model in terms of the novelty of the generated result. It is common for a model that generates melodies based on random noise to generate different results each time it generates a new result. Therefore, it is difficult to completely rule out the possibility that a good melody was generated by chance. Thus, developing a generative model for music is not always straightforward.

Considering the current situation, this study compares generative deep learning models that have achieved success in recent years, describes the characteristics of each model, and discusses which model is easier to use in which cases. In this study, we focus on recent important technologies such as LSTM, an attention mechanism, VAE, and GAN, and compare these models based on the assumption that users will use them. Although this paper also includes the generation results, we do not focus on the results but rather on the comparison of items that can be objectively stated, such as what kind of data is required to generate a melody using each model, what kind of parameter is available, and how much cost is involved in learning. For this purpose, the parameters of each model are not tuned in this study. The reason is that once we start tuning the parameters of each model, it becomes difficult to determine what and how much tuning contributed to the change in the results. Naturally, there are cases where sufficient learning has not been achieved, and the generation results presented in this study will be inferior to the results of melody generation studies using the original technique (especially the best data that can be presented to demonstrate the technique), but we will make the comparison as simple as possible while assuming this.

Recent Examples of Music Generation Research

In this chapter, we will discuss recent research on music generation, especially techniques related to deep learning. Research on music generation and automatic composition techniques has been developed since the early days of computers, and many techniques have been proposed so far. Examples of research on music generation before the penetration of deep learning technology are summarized in detail in the survey paper by Matsubara et al.4 Deep learning-based music generation technologies that have made significant progress in recent years have also been investigated in detail by Briot et al., who summarized, organized, and categorized the characteristics of each technology5. Herremans et al. also surveyed how music generation systems have

---


evolved by classifying them according to their purpose and function. As a difference between these survey papers/books and this study, our comparison focuses on deep learning techniques for melodies, rather than music generation in general. Rather than describing a specific network presented in a particular study, we will also compare the details of the key technologies that cannot be dismissed when discussing generative deep learning models. For a bird's-eye view of music generation techniques, please refer to the above survey.

In this section, we introduce some examples of research related to melody generation models that we focus on in this study. By assuming the generation of melodies, we focus, in particular, on symbolic processing-based generative models.

Roberts et al. proposed MusicVAE, which is a model for latent space representation of time-series data by a recurrent VAE using RNNs as an encoder and a decoder of the VAE. As the method is based on VAE, it does not require any training label and can be trained on the basis of whether the training data can be reconstructed successfully. In addition, the evaluation of the model can be verified by reconstruction quality. Furthermore, in the case of MusicVAE, the quality of the generated results is evaluated and verified by listening tests. In addition, the acquired latent space representation can be used for vector interpolation and arithmetic, which makes it possible to prepare two reference melodies and bring one melody to another. This will be useful in situations where a user wants to add different melodic elements to a melody that the user has already created in music production.

A GAN is arguably the most important model among existing generative deep learning techniques, and many derivatives have been proposed. A GAN is a technology that has achieved great success mainly in the field of image generation, but it has also been used in the field of music generation. MuseGAN proposed by Dong et al. achieves polyphonic multitab music generation. On the other hand, GAN-based methods require some ingenuity in processing time-series data. In the case of MuseGAN, the melody is modeled using a convolutional layer on the piano roll representation of data, which takes into account the structure of the data in close proximity but does not take into account the context of distant times. In addition, because it can only learn a predetermined number of measures, for example, it can generate music every two measures, but it cannot generate music of any length because it cannot generate music that follows the previous two measures. It is difficult to control the generation result because the GAN-based method is based on random noise. The evaluation of MuseGAN is performed by quantifying and analyzing several indices defined by the authors (such as the percentage of empty measures), as well as by user studies. Because user studies are limited to a comparison among the three models of MuseGAN, it is difficult to judge the superiority of the method over other methods. When using a model such as MuseGAN in actual music production, we can assume that the music will be generated by MuseGAN in the initial state when the music is about to be created, and then, it will be edited by a person to create the desired music. In this case, the model can be called a literal autocomposer because it is as if the music is composed at the touch of a button. If the quality of generation and the controllability of the results are improved, it may be used for applications where listeners listen to ad hoc generated music according to their mood.

An RNN is a typical model for handling time-series data such as music. Boulanger-Lewandowski et al.

---

Comparison of Deep Learning Models for Melody Generation. (Tatsunori Hirai)

proposed a music generation model combining an RNN and RBM in 2012. The prediction accuracy of RNN-based models can be evaluated, and they compare their accuracy with traditional methods such as N-GRAM and GMM+HMM. Therefore, in terms of prediction accuracy, we can confirm the improvement in accuracy of modeling, that is, the evolution of technology. On the other hand, as mentioned earlier, in music generation, a model with high prediction accuracy is not necessarily a model for generating “good melodies.”

Simple RNNs are prone to gradient explosion and gradient vanishing, and to solve this problem, gated RNN models such as LSTM and gate recurrent units (GRU) are now the mainstream. However, the problem with LSTM is that it is difficult to fully consider the influence of data at a time far from the data of interest. An attention mechanism is a solution to this problem, and the model called Transformer, which replaced an RNN with an attention mechanism, has achieved great success mainly in the field of NLP. As models using transformers in music generation, MuseNet by OpenAI and Music Transformer by Huang et al. have achieved high-quality music generation and have been the trending topic. For Music Transformer, various evaluations, including listening tests, have been reported, but for MuseNet, there was insufficient information at the time of writing this paper. In this situation, it is impossible to determine whether MuseNet or Music Transformer is the superior model. One of the interesting results reported in Music Transformer's listening test was that compared with the baseline Transformer (a different baseline model than Music Transformer) and the LSTM-based model called Performance RNN, which were included in the comparison, the LSTM-based method was rated higher than the Transformer. This is despite the fact that the baseline Transformer was superior in terms of perplexity. Although this result was not statistically significant in a one-to-one comparison between the Transformer and LSTM-based method, it shows how difficult it is to evaluate whether a melody generation model is good or bad. Although the Transformer with low perplexity is better than the LSTM-based method in terms of prediction accuracy of data, it suggests that the results of the subjective evaluation by the listening test may have been different. Most of the studies introduced here are not specialized for melody generation. Most of them are models for chord generation, and polyphonic music generation with multiple tracks. However, many of the methods can be applied to melody generation by changing the network structure and training data. Because each method has its strengths and weaknesses, different constraints, and learning conditions, it is difficult to make a simple comparison, even if we focus on the task of melody generation. For example, it may be possible to evaluate the quality of the results generated by training on the same dataset under the same conditions as possible and by conducting subjective evaluation experiments on the same group of subjects. However, depending on the situation in which the generative model is to be used, the controllability of the results may take precedence over the quality of the results, so it is not necessarily important to assign a superiority or inferiority rating to the results. Thus, evaluation of generative models is not a straightforward task.

The purpose of this study is to consider a situation in which users create melodies using generative deep learning models and to compare the features of each model. As the complexity of the network architecture increases, it becomes more difficult to determine which elements have been affected, so the models to be compared in this paper will be limited to fundamental models that are important when discussing generative deep learning.

Assumptions for the Comparison of Generative Deep Learning Models for Melodies

In this chapter, we train various generative deep learning models on existing large music datasets, and compare them by generating new melodies based on the given input melodies.

The following six methods for melody generation are specifically compared.

- Random generation
- Fully connected deep neural network
- VAE
- Seq2seq (LSTM)
- Seq2seq with attention
- GAN

Owing to the characteristics of each model, the melody generation task does not have the same conditions, but the input melodies will be identical. The melody to be generated can be a new melody following the input melody, a melody similar to the input melody, or a new melody unrelated to the input melody depending on the features of the model. Although it is possible to generate melodies using methods other than those described in this study, depending on how the data are represented and how the training is performed, it is impractical to compare all methods of melody generation, so in this study, we constructed and trained one network for each model.

As mentioned in the first chapter, it is difficult to fairly measure the effect of parameter tuning on the generation results across models. For example, it can be assumed that tuning the learning rate will significantly improve the performance of some models, whereas it will not have a significant impact on others. Therefore, we do not consider here the tuning of the hyperparameters of each model or the additional modules to improve learning. It is assumed that training and generation will be performed using the most basic network configuration possible. Although the generation results of each model can be improved by devising the structure of the model, it is difficult to measure whether the improvement in the generation results is due to the intrinsic nature of the model or due to successful parameter tuning. For this reason, we will use the basic network configuration and parameters as described in the introductory books and tutorials that explain each method. This is because we believe that excessive parameter tuning can be a bottleneck in comparing methods. In the future, we are planning to include the difficulty of parameter tuning in the comparison, but in this study, we only compare the basic network configuration. As a minimum consideration, we pay attention to indicators that determine the progress of learning, such as whether values such as loss and perplexity decrease, and avoid the situation where no learning occurs. A lower loss does not necessarily mean that the model is effective. Depending on the training task, it may be difficult to lower the loss, or it may cause overfitting when there are insufficient test data available.

In the following sections, we describe the conditions for learning and generating melodies in each model.

Dataset

In deep learning models, training data are the key. It is desirable to use a large volume of well-developed data as possible. The quality and quantity of the training data are also factors that significantly influence the final output.

In this paper, we use melody data extracted from The Lakh MIDI dataset. The Lakh MIDI dataset comprises 176,581 MIDI files collected from the Web. Because the target of this study is melody, only melody is extracted from this dataset. The extraction of melodies was performed according to the preprocessing of

---

Comparison of Deep Learning Models for Melody Generation. (Tatsunori Hirai)

the distributed representation learning method of melodies by Hirai and Sawada\textsuperscript{13}, and as a result, 10,853 melodies were obtained. In addition, the preprocessing method described above is used to estimate the tones of the acquired melodies and transpose them as necessary to unify the tones of the melodies in the dataset.

Data Representation for Melodies

A melody is composed of notes that are combinations of pitch and note value (length) and can be represented by a sequence of notes. In this study, we focus only on the sequence of notes as the target of melody generation and do not handle the velocity and other information attached to the original MIDI data. In addition, only monophonic melodies are considered for modeling.

There are several ways to represent melodies as data, but in this study, we assign unique IDs to combinations of note/rest types and pitches and treat them as vocabulary. This makes it easy to apply various methods used in the field of NLP. As a method for expressing melody data, this representation style has the same expressive power as the ABC notation method. In this research, the note number (note name) and note value of each note are obtained directly from the MIDI data and stored as array data, including information on rests. For example, a quarter note of C4 is represented as “C4:1.0,” an eighth note of D#5 as “D#5:0.5,” and a quarter rest as “R:1.0,” which are represented as text, and IDs (numbers) that use them as vocabulary are used for learning and generation. For the learning of VAEs, which will be described later, the dimensionality of the input and output data must be fixed, so the distributed representation learning method for melodies by Hirai and Sawada\textsuperscript{14} is used, and melody phrases are converted into vectors of fixed length for processing. Although there are methods based on VAE that allow for flexibility in the length of the input, only simple VAE is used for comparison in this study, so melody vectors are employed here.

Conditions for Comparison

To compare different models fairly, it is necessary to make the conditions as consistent as possible. However, it is difficult to compare models with different network structures and input/output representations in a completely fair manner. Therefore, in this study, we do not focus on specific factors, such as the results of generation, but compare as many aspects as possible. To match the conditions where possible, we use same datasets for training, and same data representation methods described in previous section for methods except VAE. In addition, as a scenario for melody generation, we assume that there is an existing input melody and a new melody is generated on this basis. This is intended for situations such as when a person is composing a song and is stuck and needs the help of a machine to complete the song. However, not all models compared in this study can generate melodies according to this condition.

For VAE and GAN, completely new melodies are generated on the basis of the input noise vector. In the case of VAE, it is possible to use the encoded value of an input melody as the value of this noise vector, and the melody vector obtained by decoding the noise vector is expected to be close to that of the input melody. Therefore, for the task of melody generation using VAE, we will generate another melody that is closest to the result of encoding and decoding an input melody. This is a model that can be used in situations where a machine is asked to suggest a different pattern of a melody that has been created, such as when a user wants to listen to a new melody that is similar to a melody that the user likes.

In the case of GAN, some methods that can be trained to produce targeted generation results similar to


those of VAE have been proposed, but because the purpose of this study is to compare melody generation for simple GAN, we generate a completely new melody based on random input noise. Therefore, in the case of GAN, melody generation is independent of the input melody.

The melody of the children's song “Twinkle Twinkle Little Star” is used as the input melody by methods other than GAN. Melody generation is performed on the basis of the input melody for the trained model, and in principle, the melody generated in the first trial is used to compare the results. We will not fine-tune the model based on results, but we will at least tune the parameters based on the learning progress to guarantee that learning will proceed.

Training/Generation of Each Model

In the following, we describe how the six models listed in previous chapter are used to train the models and generate the melody. A comparison of melody generation using the various methods described here will be described in a later chapter.

Random Generation

The purpose of this study is to compare generative deep learning models, but as a baseline, we will also focus on what results can be obtained by randomly generating melodies without using a generative model. Random generation is a method for generating a sequence of notes randomly without learning. The procedure for random generation is to prepare an array of note names (including R for rests) and an array of note values with lengths from sixteenth notes to whole notes, generate two random numbers to determine the index of the array, and combine them to randomly select notes. By repeating this process until the length of the note sequence reaches a predetermined length, a random melody can be generated. Random generation naturally produces a different melody each time it is run, and it has no effect on what kind of melody is given as an input.

It is important to decide in advance which octave notes will be selected. Otherwise, pitch fluctuations will occur. Therefore, the pitches to be generated here are limited to the octave of C4, and the bandwidth of the generated melody is limited to one octave. In this regard, more expressive random generation is possible by incorporating statistical approaches, such as sampling based on the frequency distribution of the pitch and note value in the training data. Afterward, in all models, except VAE, the pitch of the generated notes will be limited to the octave band of C4.

Fully Connected Deep Neural Network

We will also look at how melodies can be generated by so-called simple deep neural networks (DNN). Because we assume a situation in which a new melody is generated following an input melody, we constructed a network in which the input to the network is the type of note at a certain time and the output is the type of note following it. Specifically, we used one-hot vectors for both input and output using the ID representation of notes described in previous section. In addition to the input and output layers, three fully coupled layers (300 units each with sigmoid activation functions) were prepared as intermediate layers, and training was performed using the dataset introduced in previous section (data size: 3,992,505). Because the octave was limited, the pattern of note transitions was limited, and the loss values converged soon after the start of learning.

For melody generation, the ID of the last note of the input melody was inputted to the network as a one-hot vector, the output value obtained was regarded as the probability that each note would be outputted next,
Comparison of Deep Learning Models for Melody Generation. (Tatsunori Hirai)

and the next note was selected on the basis of a random number. A melody was generated by repeating it an arbitrary number of times.

**VAE**

In the case of learning the VAE model, the input data are encoded and represented as a vector in the latent space, and then, the input data are recovered by decoding the latent vector. In other words, we train the network such that the input and output data are the same. Therefore, learning with a structure similar to the previous models, where ID representations of notes are inputted and then restored, does not provide significant meaningful learning in terms of generating new melodies using VAE. Certainly, it is possible to solve such problems by constructing a model that can take time series into account, such as MusicVAE\(^{15}\), but the purpose of this study is only to verify the basic model.

To generate a melody as an output of a VAE, the input and output should both be vectors such that they represent a melody. Therefore, we used melody2vec by Hirai and Sawada\(^{16}\) to split the melody of the dataset into phrase-by-phrase segments and obtained a vector of melodies. The training of the VAE that we examine in this study is performed on this vector of melodies. Specifically, for 100-dimensional melody vectors obtained using melody2vec, we trained a network of VAE to obtain a 10-dimensional latent space using an encoder and a decoder, each consisting of two fully connected layers. The data size of the melody vectors in the training data was 967,873. The training was conducted for 500 epochs.

When using this model to generate a melody, instead of generating a melody following an arbitrary input melody, another melody closest to the melody vector obtained as an output from an arbitrary input melody was calculated on the basis of the cosine similarity and output. In other words, it searches for phrases of other melodies similar to the input melody and combines them to generate a new melody.

**Seq2seq (LSTM)**

As a seq2seq model, an encoder–decoder model based on LSTM, we constructed a network with a structure such that a note ID sequence is an input and the next note ID sequence follows it. The encoder simply concatenates the embedding layer and LSTM layer with the number of input data and outputs the vector \(h\) of hidden layers that is the output from the last LSTM cell. In the decoder, the structure is such that the received \(h\) is inputted to the LSTM cell, and the output is inputted to all coupling layers, as well as flowing to the LSTM cell at the next time. The vector \(h\) in the hidden layer received from the encoder is also inputted to all coupled layers and LSTM layer at each time to improve learning\(^{17}\).

The size of the input data is a parameter that determines how many notes are received as an input and how many notes are outputted, as well as how much past information is considered. When a dataset is divided into training and test data for validation, the smaller the input data size is, the more accuracy improves as the learning proceeds, but this merely makes the learning task easier and does not necessarily mean that effective training has been achieved. When the input data size was set to a large value, accuracy barely improved


even though the loss decreased. In this study, the input–output data size for training seq2seq was set to 5, and training was performed for 25 epochs. The number of dimensions in embedding was set to 16, and the number of dimensions in the hidden layer $h$ of LSTM was set to 256, resulting in a data size of 3,895,847.

In the melody generation phase, the melody was generated by inputting the last five notes of the input melody, estimating the following five notes, and then sequentially estimating the following five notes.

**Seq2seq (LSTM + Attention Mechanism)**

By adding an attention mechanism to seq2seq described in previous section, we introduce a context vector that enables the model to pay attention to data at distant times. We inserted an attention layer between the LSTM layer of the decoder and the fully connected layer of the network described above, modified it to receive the values of the hidden layer $h$ at all times from the encoder, and trained it under the same conditions as seq2seq in previous section.

**GAN**

GAN training was implemented in such a way that two networks performed adversarial learning: a discriminator, with a fixed number of notes and one-hot vectors for the number of notes as input, and a generator, which generates one-hot vectors for the number of notes using a fixed-length noise vector $z$ as seed. Both the discriminator and generator are composed of four fully connected layers.

The dimension of the noise vector $z$ was set to 100, and the number of notes inputted to the network at one time was set to 100. As a result, training proceeded with 10,216 melodies. Because the GAN implemented in this study cannot generate a melody considering the input melody, it generates a completely new melody based on a random input noise. Therefore, as with random melody generation, the result of melody generation is an output that is independent of the input melody.

**Comparison of Generative Models**

Here, we compare the melody generation models described in previous chapter.

**Comparison of the Features of Each Generative Model**

In this section, we will compare the characteristics of each model by listing items that can be evaluated objectively. The features of random generation are not discussed here because they are not trained, and we will mainly focus on them in later section when comparing results.

**Comparison of Network Structures**

In this section, we will focus on the comparison of the structure of each network discussed in this study. Simple DNNs are generally not used as generative models but rather as discriminators that can perform multiclass classification based on arbitrary inputs. In this study, we have chosen the simplest DNN for comparison, but it has some shortcomings such as its inability to consider time-series data in their original form. Therefore, it must be said that a simple DNN is not sufficiently effective to represent advanced time-series data such as melodies. On the other hand, various network structures and modules related to DNNs have been proposed, and by making full use of them, networks with desired characteristics can be realized, and generative deep learning models can be regarded as a derivative of simple DNNs. For example,
Comparison of Deep Learning Models for Melody Generation. (Tatsunori Hirai)

convolutional neural networks (CNN) that introduce convolutional layers have been mainly successful in the image processing field, but they also work well with piano roll representations in music, enabling us to model the structure of local sound changes. Depending on the design of the network, various applications can be realized, such as generating a continuation of a melody that matches a melody that is in the process of being created, interpolating a melody that has blank spaces in between, or generating another similar melody.

The development of generative deep learning was spurred by the development of VAE. The input data are represented as a vector in the latent space by the encoder, and the decoder learns to recover the input data from the vector. Unlike simple autoencoders (AE), in the case of VAE, data recovered from vectors in close distance in the latent space will have similar features. Owing to the structure of the networks, they are not suitable for applications such as generating a melody that follows the input melody, but they can generate a melody that resembles the input melody. Using VAE, melody morphing can be realized on the basis of the results of the interpolation of vectors in the latent space of multiple melodies.

Seq2seq is an encoder–decoder model using RNNs, which is implemented using LSTM in this study. The major advantage of using RNN is that time-series information is taken into account. For training, the series length of input/output data is fixed, but it is also possible to represent input/output of flexible length using an end-of-sentence symbol as in NLP. However, seq2seq may not be able to encode sufficient information when huge information is inputted at once because the vector size of the hidden layer output from the encoder is constant regardless of the length of the input data. Therefore, by introducing the attention mechanism, the hidden layer vector $h$ output from all LSTM cells can be passed to the decoder, which can then decode the vector based on the encoded information from all input states. Furthermore, by flexibly changing the factors of interest at each time, the data can be fully influenced by data at distant times. In the case of the two seq2seq models presented in this study, the input–output data series length was set to 5, so the model was trained to estimate the next five notes from the input of five notes. By increasing the length of the series, inferences can be made on the basis of the information at more distant times but at the cost of longer learning times. Depending on the training method, the model can be used for various applications such as melody style conversion. In the application of generating a melody following an arbitrary melody, as described in this study, the model can be used to suggest a candidate to a user who is stuck in creating a melody.

A GAN is an important model that has rapidly accelerated the development of generative deep learning techniques, and there are many derived models. The model is such that adversarial learning is performed between a generator, which generates melodies based on input noise vectors, and a discriminator, which discriminates between the generated melodies and real melodies. For melody generation, a noise vector is inputted to the trained generator to generate a new melody. In this case, the result is generated from zero, which can be used to generate a new melody, regardless of the previous production work, but it is difficult to generate a melody that matches the music that a user is currently creating. Although many methods have been proposed to control the generation result by the generator by providing conditions, one of the characteristics of GAN is that controlling the generation result is difficult. Another problem with the task of melody generation is that the length of the data needs to be fixed. In the case of image generation, fixing the size of the data is not a challenge, but in the case of melodies, having a fixed length is not desirable.

As mentioned above, each model has its characteristics, but in common, because the structure of a deep learning model can be rearranged, there is room for some degree of customization to suit the purpose, and this is the reason why various generative models are constantly being proposed on the basis of each method. For example, the various models presented in this study cannot generate a melody by considering the length of one measure, but they can generate a melody for each measure by changing the input/output data representation method.
Comparison of Time Required for Training

The time required for training depends largely on the execution environment and the implementation method. Therefore, it is a difficult factor to make accurate comparisons. However, in this study, it is possible to make relative comparisons because all models are implemented in the same computing environment. Although we are attempting to implement the system in such a way as to minimize the difference in processing time, the difference in processing time caused by the difference in modules used in the detailed part of the implementation may be accumulated in each iteration. Therefore, note that the data presented here are not exact but are for reference only. The time required for learning also varies significantly depending on the parameters. The data shown here are based on the time taken to train one epoch under the conditions described in the previous chapter.

Table 1 shows the ratio of the time required to train each model when the time required to train one epoch of a DNN is 1. Note that a simple comparison is difficult because the dimensionality of the data input and output to the network and parameters to be trained are different. According to the results, the time required for learning increases as the number of components of the network increases, from VAE < GAN < DNN < seq2seq < seq2seq with the attention mechanism. Although this result is within the expected range, we can say that it is necessary to construct a network with a simple configuration for fast training. For example, if the benefits of adding an attention mechanism are insufficient to justify the increase in learning time, the option of not adding an attention mechanism may well be considered. In particular, in cases where the generative model is trained by repeated trial and error while changing the training data, the shorter the training time is, and the more beneficial it is to be able to perform various trials.

<table>
<thead>
<tr>
<th>Model</th>
<th>DNN</th>
<th>VAE</th>
<th>Seq2seq</th>
<th>Seq2seq with an attention mechanism</th>
<th>GAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing time</td>
<td>1.0</td>
<td>0.0095</td>
<td>2.1</td>
<td>5.8</td>
<td>0.072</td>
</tr>
</tbody>
</table>

Training Difficulty

In this section, we will list items that occurred when we implemented and learned each method, rather than objective indicators, in bullet-point form. Although it is uncommon for technical studies to describe perspectives such as implementation difficulty and learning difficulty, we describe them in this study as one perspective for comparing each model. Note that depending on the implementation method, the phenomena listed here may not necessarily apply.

- Losses during training of the DNN, seq2seq, and seq2seq with an attention mechanism decreased to a certain extent at the first epoch, and then, they did not significantly decrease after that and oscillated within a certain range.
- Loss of VAE continued to decrease even as the number of epochs was increased (attempted up to 500 epochs).
- GAN training was unstable, and in many cases, the training proceeded in such a way that the output of the generator was all zero.
• The results of the melody generation by the GAN often resulted in the same melody even when the input noise was changed.

The above issues arose during the implementation of the model presented in this study. Although several methods have been proposed to improve the training problem in the GAN, the improvement is beyond the scope of this study.

Comparison of Generated Results

In this study, we do not aim to compare the results of melody generation. The reason for this is that even if there is some significant difference between the methods (or even if there is none) when they are compared using an evaluation method with low reproducibility based on a few generated cases, this is no sufficient basis to conclude the superiority of each model. Nevertheless, because we trained the models based on the goal of generating melodies and obtained the results of actual melody generation, we present them here only as reference data. Note that the results shown here are melodies generated without tuning various parameters and thus do not show the full potential of each method. In addition, the models that use random values to generate melody produce different results for each trial. This is especially true in the case of random generation, but some other methods produce different results for each trial. As a result, it should be noted that there is a possibility that by increasing the number of trials, a good melody may be generated by chance, or conversely, a melody that cannot be called music may be generated by chance.

Here, the number of trials is limited to three for each model, and the results shown are the melodies generated in the first trial. The input melody used other than random generation and generation by the GAN is the melody of the song “Twinkle Twinkle Little Star.”

The results generated by each model are shown in Figures 1–6. Here, the result of the VAE is another melody that is considered similar to the input melody, the result of random generation and the GAN is a melody that has nothing to do with the input melody, and the other melodies are melodies generated as a continuation of the input melody. In the case of the DNN, only the last note of the input melody is considered, whereas the two types of seq2seq consider the last five notes and generate the melody that follows it.

Because all melodies used in the training were transposed to the key of C major or A minor, it is natural to output notes with few #. However, in some models, many # appear in the generated melodies. In terms of rhythm, there are several unnatural long tones, irregular rhythms, etc., in the output from all models. This may simply be due to inadequate parameter tuning of each model and unstable output results, but it also indicates that when trying to implement a melody generation model, it is difficult to find a method that can successfully learn and obtain ideal generation results in a small number of trials and it is common to generate melodies not suitable for music.

The length of the melody to be generated is a factor that is difficult to control, and in this case, the process is such that notes are generated continuously until the length of the specified number of measures is exceeded. For this reason, for a model such as seq2seq, where the melody is outputted in five-note units, the generation result is cut off in the middle. For the GAN, the number of notes can be determined, but the length of each note is unknown until it is generated, so the result shown here is the generated melody of 100 notes cut in the middle. For the VAE, because the generation is based on a melody vector that has nothing to do with the length of the melody, the length of the output melody cannot be controlled, and a melody with a different length is generated for each trial.

Figure 1  Example of melody generation by random generation
Consideration of Scenarios for Using Each Model in Music Production

The true value of a generative model can be demonstrated only when it is used in actual music production and appreciation. In this section, we examine situations in which the various models implemented in this study can be used.

First, the DNN model is designed to infer the next note based on the current note information. When using this model in actual music production, it is possible to listen to the results generated by the model and decide the next note when you are not sure what to choose.

For melody generation using the VAE, we used the VAE to generate another melody that is similar to the input melody, rather than generate a completely new melody. This method can be used by listeners to find another melody similar to the one they like or by creators to use a melody with a different pattern than the one they had in mind. A more advanced application would be to directly search the latent space of the VAE to find a melody that is located between multiple melodies or to create a melody that is closer to one melody than to another.

For seq2seq, because the output is a coherent sequence of notes, it can be used to suggest a candidate melody to follow the current one when a creator is stuck in creating a melody.

For melody generation using the GAN, the model generates a completely new melody based on random noise vectors, so it can be used in situations where a user wants to mass-produce melodies, but such situations are uncommon. In the future, if the accuracy of melody generation by a GAN is improved so that it can always generate a “good melody” in terms of audibility, it may be used for applications that enable users to continue listening to music for a limited time, which is tolerable for appreciation.
Another Direction in Music Generation Research

The current mainstream of music generation research is related to generative deep learning models. On the other hand, it is also important to study interfaces that examine how to apply these technologies to support creation. Zhou et al. proposed a system that efficiently searches for a user's preferred melody in a high-dimensional latent space using human-in-the-loop optimization\textsuperscript{18}. Dinculescu et al. have proposed MidiMe, a method to personalize the MusicVAE model based on user data\textsuperscript{19}. Such technologies are expected to make it easier for end-users to access the results of automatic composition. Beyond that, we can assume that there will be creators who want to design a method for acquiring latent space. For example, it would be natural for creators to demand that they choose the training data for building their original models. Currently, Magenta.js\textsuperscript{20}, a JavaScript API, enables the development of applications that use the models used in Magenta, but it is unsuitable for end-users such as creators because it requires programming. Magenta Studio has been proposed as an interface that can use the API provided by Magenta.js as a VST plug-in for the DAW software such as Ableton Live\textsuperscript{21}. However, although Magenta Studio improves usability by creators, it is designed in such a way as to discard the possibility of adjusting the details of training models. Similarly, although ORB Producer Suite and Flow Machines are being commercialized as tools that enable creators to use the generated models, no system that allows creators to freely design generated models has yet been proposed. Thus, at present, there is a gap between creators who want to use automatic composition technology and engineers who develop automatic composition technology. The tuning of a model by an engineer has a large impact on results, there is not much room for creators to reflect their creativity on a generated model, and they are limited to using tools provided by engineers. As a future direction of music generation research, it is necessary to consider how to use generative models to realize a composition by humans and artificial intelligence (AI), and examples of such research are being published\textsuperscript{22}.

We believe that in the same way that John Cage invented the prepared piano and pursued his musical expression by going into the instrument, we need to develop a system that enables creators to create their generative models. In the direction of such technology, we can imagine a future in which engineers could publish new expressions. Similar to the invention of the theremin by Lev Theremin or the invention of the synthesizer by Robert Moog, we can expect that the invention of a new expression technology will become an expression, and generative deep learning technology has such potential. At present, the task of proposing a new generative deep learning model and optimally adjusting the parameters of that model requires a great deal of effort. This process requires a significant number of trial and error that it is not an exaggeration to say that it is an activity for new musical expression. If we can develop a generation technology that enables creators to perform this task interactively, we believe that these generative technologies will be more widely used in music production.

To achieve this, it is necessary to solve various problems in constructing new generative models and lower the threshold for their introduction. The current music generation model has various issues in terms of


technology, time, convenience, and controllability. For example, if you compare it to an ordinary synthesizer, the difference is huge. In the case of a synthesizer, adjusting the knob that controls the parameter and pressing a key on the keyboard immediately produce new sound feedback, but in the case of parameters in a deep learning model, adjusting the knob and retraining for several hours or more finally produce new music. We expect that these issues will be addressed by waiting for the emergence of architectures capable of faster training and efficient and intuitive fine-tuning methods, as well as the development of end-user-friendly interfaces that reflect them. In addition, it is difficult for end-users to understand the principles of complex generative deep learning models, and they end up using them in a black-box manner without sufficient technical understanding. There is a lot of discussion regarding coexisting with AI, but we cannot create with AI unless we understand the tools. The other problem is that many existing music production tools have deterministic behavior in which the output is uniquely determined for each parameter value, but the output of many generative models varies because of stochastic behavior. Therefore, rather than setting parameters with a creator's intention in mind, the current usage is to change parameters and adopt the result that the creator likes by chance. To realize a future where the results of music generation by generative models are introduced to music production sites as more practical, it is desirable to realize a system with fewer black-box processing processes as this reduces the uncertainty of the results as much as possible.

Conclusion

In this study, we compared melody generation by various generative deep learning models that have achieved success in recent years. We have discussed DNN, VAE, seq2seq (LSTM), an attention mechanism, and GAN, but many technologies can be compared, and the ones we have focused on are only part of them. This comparison focuses only on basic but important techniques that are indispensable among recent generative deep learning models. Although we could not compare the merits and demerits of each model clearly, we have summarized the characteristics and differences of each model as objectively and multilaterally as possible. In the future, we would like to develop a method that can fairly evaluate the quality of melodies generated by each model. In addition, we would like to consider a framework that would enable us to evaluate the most advanced generation technologies that were excluded in the evaluation in this study.

A “good melody” is difficult to define as it is perceived differently by different people. Therefore, there are limitations to the data-driven approach, and even if it is possible to generate “not bad melodies” that do not deviate from music theory, there will be a large gap between them and “good melodies.” Therefore, what is important is to realize an interface that enables each user to access melodies that they consider “good melodies.” It is difficult to find a “good melody” by blindly searching the melody space. However, using recently developed generative deep learning techniques, we can expect to acquire latent spatial representations that make it easier to search for a “good melody” and realize interfaces that can approach the desired melody as efficiently as possible. In the future, we would like to discuss guidelines for determining the direction in which the technology should go, which are necessary for realizing such technology.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Numbers JP19K20301.