MELODY GENERATION FOR POP MUSIC VIA WORD REPRESENTATION OF MUSICAL PROPERTIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Automatic melody generation for pop music has been a long-time aspiration for both AI researchers and musicians. However, learning to generate euphonious melody has turned out to be highly challenging due to a number of factors. Representation of multivariate property of notes has been one of the primary challenges. It is also difficult to remain in the permissible spectrum of musical variety, outside of which would be perceived as a plain random play without auditory pleasantness. Observing the conventional structure of pop music poses further challenges. In this paper, we propose to represent each note and its properties as a unique '*word*,' thus lessening the prospect of misalignments between the properties, as well as reducing the complexity of learning. We also enforce regularization policies on the range of notes, thus encouraging the generated melody to stay close to what humans would find easy to follow. Furthermore, we generate melody conditioned on song part information, thus replicating the overall structure of a full song. Experimental results demonstrate that our model can generate auditorily pleasant songs that are more indistinguishable from human-written ones than previous models. [1]

1 INTRODUCTION

Recent explosion of deep learning techniques has opened up new potentials for various fields of multimedia. Vision and language have been its primary beneficiary, particularly with rising interest in generation task. Considerable amount of recent works on vision and language have hinged beyond mere generation onto artistic aspects, often producing works that are indistinguishable from human works (Goodfellow et al. (2014); Radford et al. (2016); Potash et al. (2015)). On the other hand, it is only recently that deep learning techniques began to be applied to music, and the quality of the results are yet far behind those in other domains, as there are few works that demonstrate both euphonious sound and structural integrity that characterize the human-made musical contents. This unfortunate status holds true for both music in its physical audio format and its abstraction as notes or MIDI (Musical Instrument Digital Interface).

Such lagging of deep learning-enabled music generation, particularly in music as abstraction, can be attributed to a number of factors. First, a note in a musical work contains various properties, such as its position, pitch, length, and intensity. The overall tendency of each property and the correlation among them can significantly vary depending on the type of music, which makes it difficult to model. Second, the boundary between musical creativity and plain clumsiness is highly indefinite and difficult to quantify, yet exists. As much as musical creativity cannot be limited, there is yet a certain aspect about it that makes it sound like (or not sound like) human-written music. Finally, music is not merely a series of notes, but entails an overall structure of its own. Classical music pieces are well-known for their high structural complexity, and much of pop music follows the general convention of verse - pre-chorus - chorus structure. This structure inevitably necessitates different modeling of musical components; for example, notes in the chorus part generally tend to be more high-pitched. It goes without saying that these structure-oriented variations further complicate the modeling of music generation.

[1] Code and dataset will be publicly available prior to the publication of this paper. Check the following URL for our demos: https://drive.google.com/open?id=0B7FM2yuGVreASmw3b2YxM2xtdUk

In this paper, we propose a new model for music generation, specifically symbolic generation of melodies for pop music in MIDI format. The term "*pop music*" can have different meanings depending on the context, but we use the term in this paper to refer to its musical characteristics as conventionally accepted. Specifically, it refers to the songs of relatively short lengths, mostly around 3 minutes, with simple and memorable melodies that have relatively low structural complexity, especially in comparison to classical music. Music in MIDI format (or, equivalently, in notes) can be considered a discrete abstraction of musical sound, analogous to the relationship between text and speech. Just as understanding text is not only essential in its own merit, but provides critical clues to speech and language in general, understanding music at its abstraction can provide an ample amount of insights as to music and sound as a physical format, while being fun and significant *per se*.

We address each of the challenges described above in our proposed model. First, we propose to treat a note and its varying properties as a unique 'word,' as opposed to many previous approaches that took each property into consideration separately, by implementing different layers for generation. In our model, it suffices to train only one model for generation, as each 'word' is an incarnation of all of its properties, thus forming a melody as a 'sentence' consisting of those notes and the properties. This approach was inspired by recent successes in image captioning task (Karpathy & Li (2015); Vinyals et al. (2015); Xu et al. (2015)), in which a descriptive sentence is generated with one word at a time in a recurrent manner, while being conditioned on the image features. Likewise, we generate the melody with one note at a time in a recurrent manner. The difference is that, instead of image features obtained via convolutional neural networks (CNN), we condition the generation process on simple two-hot vectors that contain information on chords sequences and the part within the song. Chord sequences and part annotations are automatically generated using multinomial hidden markov model (HMM) whose state transition probabilities are computed from our own dataset. Combining Bayesian graphical models with deep neural netweorks (DNN) has become a recent research interest (Gal & Ghahramani (2016)), but our model differs in that HMM is purely used for feature input generation that is processed by neural networks.

Second, we enforce regularization policy on the range of notes. Training with a large amount of data can lead to learning of excessively wide range of pitches, which may lead to generation of melodies that are not easy to sing along. We alleviate these problem by assigning a loss function for the range of notes. Finally, we train our system with part annotation, so that more appropriate melody for the corresponding part can be generated, even when the given chord sequences are identical with other parts of the song. Apart from the main model proposed, we also perform additional experiments with generative adversarial networks (Goodfellow et al. (2014)) and with multi-track songs.

Our main contributions can be summarized as following:

- proposal of a model to generate euphonious melody for pop music by treating each note and its properties as single unique "word", which alleviates the complexity of learning
- implementation of supplementary models, such as chord sequence generation and regularization, that refine the melody generation
- construction of dataset with chord and part annotation that enables efficient learning and is publicly available.

2 Related Works

Most of the works on automatic music composition in the early days employed rule or template-based approach (Jacob (1996); Papadopoulos & Wiggins (1999)). While such approaches made important contributions and still continue to inspire the contemporary models, we mainly discuss recent works that employed neural networks as a model of learning to compose music, to make close examination and comparison to our model.

DeepBach (Hadjeres & Pachet (2017)) aims to generate Bach-like chorale music pieces by employing pseudo-Gibbs sampling. They discretize time into sixteenth notes, generating notes or intervals at each time step. This marks a contrast to our model that does not have to be aware of each discrete time step, since positional information is already involved in the note representation. They also assume that only one note can be sung per instrument at a given time, dividing chords into different layers of generation. On the other hand, our model can handle multiple notes at the same position, since sequential generation of notes does not imply sequential *positioning* of notes. As we will see in

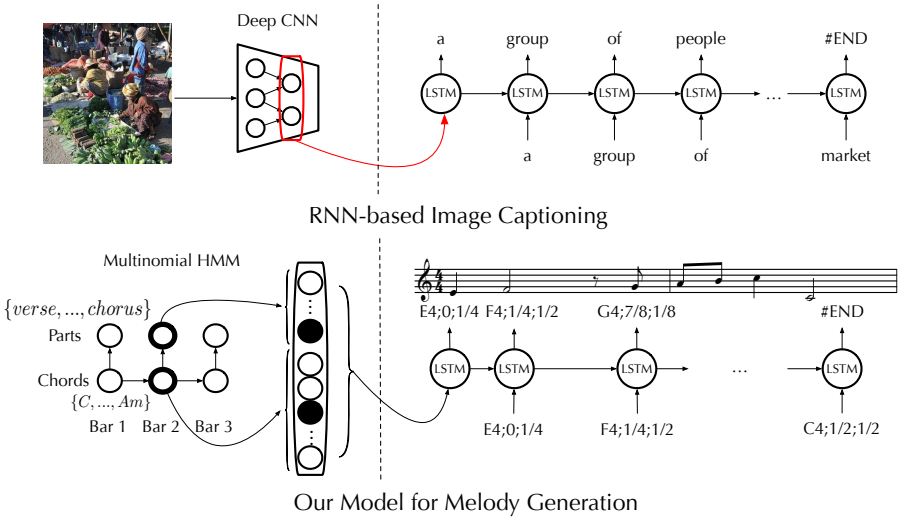


Figure 1: Visual analogy between image captioning task and our model. By grouping a note and its properties as a '*word*', we generate melody as a '*sentence*'.

Section 4.3, our model can generate simultaneous notes for a single instrument. Huang et al. (2017) also take a similar approach of applying Gibbs sampling to generate Bach-like chorale music, but mostly share the same drawbacks that make a contrast to our model.

Jaques et al. (2017) proposed RL Tuner to supplement recurrent neural networks with reinforcement learning by imposing cross-entropy reward function along with off-policy methods from KL control. *Note RNN* trained on MIDI files is implemented to assign rewards based on the log probability of a note given a melody. They defined a number of music-theory based rules to set up the reward function. Our model, on the other hand, does not require any pre-set rules, and the outcome can be easily controlled with simple regularizations.

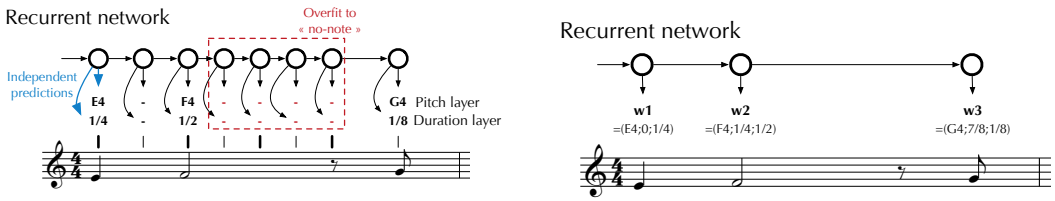
Chu et al. (2017) proposed a hierarchical recurrent neural network model to produce multi-track songs, where the bottom layers generate the melody and the higher levels generate the drums and chords. They built separate layers for pitch and duration that generate an output at each time step, whereas our model needs only one layer for pitch and duration and does not have to be aware of time step. They also conditioned their model on scale types, whereas we condition our model on chord sequence and part information.

While generating music as physical audio format is out of scope of this paper, we briefly discuss one of the recent works that demonstrated promising results. Originally designed for text-to-speech conversion, WaveNet (van den Oord et al. (2016)) models waveform as a series of audio sample x_t conditioned on all previous timesteps, whose dependence is regulated by causal convolutional layers that prevent the violations in ordering. When applied to music, it was able to reconstruct the overall characteristics of corresponding music datasets. While only for a few seconds with frequent inconsistency, it was able to generate samples that often sound harmonic and pleasant.

3 Generation Model

3.1 Melody Repereseentation

Our model for melody generation can be best illustrated by making an analogy to image captioning task. In image captioning, most popular model is to generate each word sequentially via recurrent networks such as long short-term memory (LSTM) (Hochreiter & Schmidhuber (1997)), conditioned on the image representation. In our model, we treat each note and its properties as a unique 'word,' so that melody becomes the 'sentence' to be generated. In other words, a pitch p_i with duration l_i located at t_i within the current chord sequence will be represented as a single word $w_i = (p_i, t_i, l_i)$. Accordingly, a melody will be a sequence of *words*, $s_j = (w_0, ..., w_{m_i}) \in S$. While we also use LSTM for word generation part, we condition it on music-relevant feature $x_i \in X$, instead of CNN



(a) Independent representation of each property at identical time intervals (previous)

(b) Timestep-independent word representation of multiple properties (proposed)

Figure 2: Comparison of our model for musical representation with previous model. Most of the previous models used a frame-level time granularity, which can easily lead to a model over-fitting on the repetition of the previous time step. Our proposed model of word representation alleviates this problem by encoding the time information (duration and position).

image features; namely, chord-sequence x_{chord_i} and part annotation x_{part_i}. Thus, we perform a maximum log likelihood estimation by finding the parameters set θ^* such that

$$\theta^* = \arg\max_{\theta} \sum_{(X,S)} \log p(s_i|x_i;\theta) = \arg\max_{\theta} \sum_{1}^{N} \log p(w_0, ..., w_{m_i}|x_{chord_i}, x_{part_i};\theta) \quad (1)$$

where N is the number of training samples. Figure 1 makes a visual analogy between image captioning task and our model.

Our model of melody representation makes a strong contrast with widely used approach of implementing separate layers for each property as described in Section 2. Previous approach essentially treats every 1/16 segment equally. Because this approach encounters a substantial number of segments that are repeated over several time steps, it is very likely that a statistically trained model will simply learn to repeat the previous segment, particularly for segments with no notes. It also complicates the learning by having to take the correlations among the different properties into consideration. On the other hand, our model does not have to consider intervals that do not contain notes, since our word representation already contains positional information. This puts us at advantage particularly when simultaneous notes are involved; even though notes are generated sequentially, they can be positioned at the same position, forming chords, which is difficult to implement with previous models based on time step. It also suffices to implement only one layer of generation, since the representation contains both pitch and length information. Moreover, considering pitch, position, and its length simultaneously is more concurrent with how humans would write melodies (Levitin (2006)). Visual description of our model and previous model is shown in Figure 2.

Melody generation through outputting a sequence of '*words*' is performed by LSTM with musical input features that will be described in Section 3.2. Following Karpathy & Li (2015), word vectors were randomly initialized. We used the conventionally used gate functions for LSTM as following:

$$\begin{aligned} i_t &= \sigma(W_{i_x} x_t + W_{i_h} h_{t-1} + b_i) \\ f_t &= \sigma(W_{f_x} x_t + W_{f_h} h_{t-1} + b_f) \\ o_t &= \sigma(W_{o_x} x_t + W_{o_h} h_{t-1} + b_o) \\ g_t &= tanh(W_{g_x} x_t + W_{g_h} h_{t-1} + b_g) \end{aligned} \quad (2)$$

where σ indicates sigmoid function for non-linearity activation, h_{t_1} is the memory output from the previous timestep that is fed back to LSTM, b_i is bias, and i_t,f_t,o_t correspond to input, foget, output gates respectively.

3.2 Chord Sequence & Part Generation

Since our melody generation model is conditioned on musical input features, namely chord sequence and part information, we now examine how to automate the input generation part. We employ two-fold multinomial Hidden Markov Model (HMM), in which each chord and each part is a state whose state transition probabilities are computed according to our dataset. It works in a two-fold way, in which chord states are treated as latent variables whose transitions are dependent on the part states

Algorithm 1 Regularization for pitch range

Inputs: W =initially empty weight matrix, P = softmax predictions, S = generated melody with pitches $(p_0, ..., p_n)$, pre-set minimum and maximum pitches p_{min} and p_{max}, coefficient μ
for p_i **in** S
  if $p_i > p_{min}$
    append $\max(p_i - p_{max}, 0)$ to W
  else
    append $p_{min} - p_i$ to W
Sum up the products of P and W to get cost $C = \sum W_j P_j$
Compute derivative $\frac{dE}{dp_i} = P_i(W_i - C)$
Update softmax cost by adding $\frac{dE}{dp_i}\mu$

Table 1: List of chord sequences over 2 continuous bars used in our dataset. Scale for all sequences has been adjusted to C Major.

Chord Sequences
(C-Em), (A#-F), (Dm-Em), (Dm-G), (Dm-C), (Am-Em), (F-C), (F-G), (Dm-F), (C-C), (C-E), (Am-G), (F-F), (G-G), (Am-Am), (Dm-Dm), (C-A#), (Em-F), (C-G), (G#-A#), (F-Am), (G#-Fm), (Am-Gm), (F-E), (Dm-Am), (Em-Em), (G#-G#), (Em-Am), (C-Am), (F-Dm), (G#-G), (F-A#), (Am-G#), (C-D), (G-Am), (Am-C), (Am-A#), (A#-G), (Am-F), (A#-Am), (E-Am), (Dm-E), (A-G), (Am-Dm), (Em-Dm), (C-F#m), (Am-D), (G#-Em), (C-Dm), (C-F), (G-C), (A#-A#), (Am-Caug), (Fm-G), (A-A), (F-Em)

as observed variables. Thus,

$$p(x_1, ..., x_m, z_1, ..., z_N) = p(z_1) \prod_{n=2}^{N} p(z_N|z_{N-1}) \prod_{n=1}^{N} p(x_n|z_N) \tag{3}$$

where x_i are part states and z_i are chord states. Viterbi algorithm was used for decoding.

3.3 Regularization

Training with a large amount of data can lead the learning process to encounter a wide range of pitches, particularly when scale shifts are involved in the training data as in Chu et al. (2017) or in our dataset. Such problem can lead to generation of unnatural melody whose pitch range deviates from what would be expected from a single song. We enforce regularization on the pitch range, so that the generated melody stays in a pitch range that humans would find easy to follow. We assign regularization cost to the learning process, so that a penalty is given in proportion to the absolute distance between the generated note and the nearest note in the pre-determined pitch range. Algorithm 1 describes the procedure of our regularization on pitch range, whose outcome will be back-propagated to get gradients. We set minimum and maximum pitch as 60 (C4) and 72 (C5) respectively, but it can be easily adjusted depending on the desirable type of song or gender of target singer. We set regularization coefficient as 0.0001.

4 Experiment

4.1 Setting

We collected 46 songs in MIDI format, most of which are unpublished materials from semi-professional musicians. Unofficial MIDI files for published songs were obtained on the internet, and we were granted the permission to use the songs for training from the organization owning the copyrights of the corresponding songs. It is very common in computer vision field to restrict a task to a certain domain so that the learning becomes more feasible. We also restricted our domain to pop music of major scale to make the learning more efficient.

Some of the previous works (Hadjeres & Pachet (2017)) have employed data augmentation via scale shift. Instead, we adjusted each song's scale to C Major, thus eliminating the risk of mismatch between scale and generated melody. This adjustment has a side effect of widening the pitch range of melody beyond singable one, but this effect can be lessened by the regularization scheme over pitch range as described in Section 3.

Table 2: Statistics of our dataset.

# songs	# samples	avg # notes	max # notes	min # notes	std dev
46	1912	9.33	24	1	3.60
min pitch	**max pitch**	**median pitch**	**min length**	**max length**	**median length**
53	86	69	1/16	1	1/8

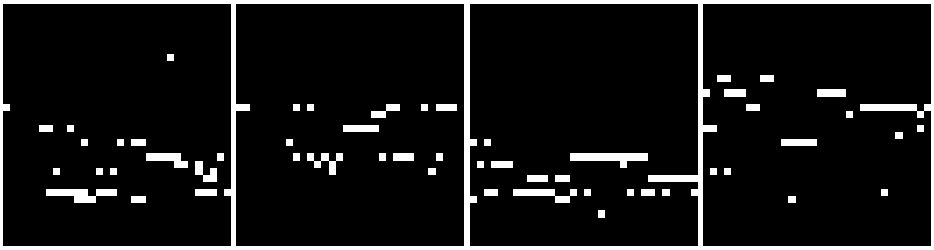


Figure 3: Visualization of songs generated with GAN.

We manually annotated chord and part for each bar in the songs collected. We restricted our chord annotation to only major and minor chords with one exception of C augmented [2]. Note, however, that this does not prevent the system from generating songs of more complex chords. For example, melodies in training data that are conditioned on C Major still contain notes other than the members of the conditioning triad, namely C, E, and G. Thus, our system may generate a non-member note, for example, B, as part of the generated melody when conditioned on C Major, thus indirectly forming C Major 7th chord. Part annotation consisted of 4 possible choices that are common in pop music structure; verse, pre-chorus, chorus, and bridge.

We experimented with n=1,2,4 continuous bars of chord sequences. Conditioning on only one bar generated melody that hardly displays any sense of continuity or theme. On the other hand, using chord progression over 4 bars led to data sparsity problem, which leads to generated songs simply copying the training data. Chord sequences over 2 bars thus became our natural choice, as it was best balanced in terms of both thematic continuity and data density. Check our demo for example songs conditioned on n=1,2,4 continuous bars. We annotated *non-overlapping* chord sequences only; for example, given a sequence C - Am - F - G, we sample C - Am and F -G, but not the intermediate Am - F. This was our design choice to better retain the thematic continuity. As for the length of notes, we discretized by 16 if the length was less than 1/2, and by 8 otherwise.

Table 2 shows some of the statistics from our dataset. Throughout our dataset construction, pretty-midi [3] framework was used to read, edit, and write MIDI files. Our dataset is publicly available with permissions from the owners of the copyright.

We ended up having 2082 unique 'words' in our vocabulary. Learning rate was set to 0.001. Total number of learnable parameters was about 1.6M, and we applied dropout (Srivastava et al. (2014)) with 50% probability after encoding to LSTM.

4.2 Evaluation

We make comparison to some of the recent works that employed deep learning to generate music in MIDI format. We performed two kinds of human evaluation tasks on Amazon Mechanical Turk, making comparison between outcomes from our model and two baseline models; Chu et al. (2017) and Jaques et al. (2017). We deliberately excluded Hadjeres & Pachet (2017) as it belongs to a different domain of classical music. In task 1, we first asked the participants how much expertise they have in music. We then played one song from our model and another song from one of the baseline models. After listening to both songs, participants were asked to answer which song has melody that sounds more like human-written one, which song is more well-structured, and which one they like better. In task 2, we performed a type of Turing test (Turing (1950)) in which the participants were asked to determine whether the song was written by human or AI.

[2]Note that, since all the songs have been adjusted to C major scale, we are using the tabular notation with root notes for convenience, instead of the conventional Roman numerals that are scale-invariant.

[3]https://github.com/craffel/pretty-midi

Table 3: Results from evaluation task 1. Numbers indicate the proportion in which our model was preferred over the baseline model.

vs. Model	Expertise	Melody	Structure	Preference	Overall
vs. Chu et al. (2017)	low	.598	.545	.542	.576
	middle	.719	.692	.619	.684
	high	.687	.712	.712	.704
	all	.691	.684	.654	.678
vs. Jaques et al. (2017)	low	.583	.458	.583	.542
	middle	.570	.427	.567	.521
	high	.598	.511	.565	.558
	all	.577	.447	.567	.530

Table 4: Results from evaluation task 2. Deception rate indicates the proportion in which the song was believed to be made by human.

Model	Ours	Chu et al. (2017)	Jaques et al. (2017)	Human
Deception rate	.680	.599	.620	.777

Table 3 shows the results from task 1 for each question and each expertise level of the participants. 973 workers participated. Against Chu et al. (2017), our model was preferred in all aspects, suggesting our model's superiority over their multi-layer generation. Against Jaques et al. (2017), our model was preferred in all aspects except in structure. Lower score in structure is most likely due to their musical formality enabled by pre-defined set of theoretical rules. Yet, our model, without any pre-defined rule, was considered to have more natural melodies and was more frequently preferred. Interestingly, even when participants determined that one song has more human-like melody with clearer structure, they frequently answered that they preferred the other song, implying that humanness may not always correlate to musical taste. χ^2 statistic is 75.69 against Chu et al. (2017) and 31.17 against Jaques et al. (2017), with *p*-value less than 1e-5 in both cases. Against either baseline model, people with intermediate or high expertise in music tended to prefer our model than those with low expertise. Table 4 shows the results from task 2. 986 workers participated. Understandably, songs actually written by humans had the largest proportion of being judged as humans. Our model had the best deception rate among the artificial generation models. Consistency of the results with task 1 implies that generating natural melody while preserving structure is a key for human-like music generation.

4.3 Additional Experiments

Generative Adversarial Networks (GANs) (Goodfellow et al. (2014)) have proven to be a powerful technique to generate visual contents, to the extent where the generated results are frequently indistinguishable from human-made contents or actual pictures (Radford et al. (2016); Reed et al. (2016)). Since the musical score can be regarded as a one-dimensional image with the time direction as the x axis and the pitch as the channel, we hypothesized that GAN may be able to generate music as *image*.

GANs consist of a generator G and a discriminator D. The generator G receives random noise z and condition c as inputs, and outputs contents $G(z, c)$. The discriminator D distinguishes between real data x in the dataset and the outputs of the generator $G(z, c)$. The discriminator D also receives the condition c. D is trained to minimize $-\log(D(x, c)) - \log(1 - D(G(x, c), c))$ while G is trained to minimize $-\log(D(G(x, c), c))$. We used the two-hot feature vector described in Section 3 as condition c. We used down-sampling & up-sampling architecture, Adam optimizer (Kingma & Ba (2015)), and batch normalization (Ioffe & Szegedy (2015)) as suggested in Radford et al. (2016).

Listening to the generated results, it does have its moments, but is frequently out of tune and the melody patterns sound restricted. GAN does have advantage particularly with chords, as it can 'visually' capture the harmonies, as opposed to sequential generation in our proposed model, or stacking different layers of single notes as in Hadjeres & Pachet (2017). Also, melody generation with GAN can potentially avoid the problem of overfitting due to elongated training. On the other hand, the same melody frequently appears for the same input. This is likely due to the problem known as GAN's mode collapse, in which the noise input is mostly ignored. In addition, it is difficult to know whether a line corresponds to a single note or consecutive notes of smaller lengths.

Many of the problems seem to fundamentally stem from the difference in modalities; image and music. See Figure 3 for visualization of the songs generated with GAN.

We also examined generating other instrument tracks on top of the melody track using the same model. We extracted bass tracks, piano tracks, and string tracks from the dataset, and performed the same training procedure as described in Section 3. Generated instruments sound fairly in tune individually, confirming that our proposed model is applicable to other instruments as well. Moreover, we were able to generate instrument tracks with simultaneous notes (chords), which is difficult to implement with previous generation model based on time step. However, combining the generated instrument tracks to make a 4-track song resulted in dissonant and unorganized songs. This implies that generating a multi-track song requires a more advanced model for learning that reflects the interrelations among the instruments, which will be our immediate future work. Check our demo for songs generated with GAN and multi-track song generated with our model.

4.4 Discussion

Although our model was inspired by the model used in image captioning task, its task objective has a fundamental difference from that of image captioning. In image captioning task, more resemblance to human-written descriptions reflects better performance. In fact, matching human-written descriptions is usually the evaluation scheme for the task. However, in melody generation, resembling human-written melody beyond certain extent becomes *plagiarism*. Thus, while we need sufficient amount of training to learn the patterns, we also want to avoid overfitting to training data at the same time. This poses questions about how long to train, or essentially how to design the loss function. We examined generations with parameters learned at different epochs. Generated songs started to stay in tune roughly after 5 epochs. However, after 20 epochs and on, we could frequently observe the same melodies as in the training data, implying overfitting (check our demo). So there seems to exist a '*safe zone*' in which it learns enough from the data but not exceedingly to copy it. Previous approaches like Jaques et al. (2017) have dealt with this dilemma by rewarding for following the pre-determined rules, but encouraging off-policy at the same time. Since we aim for learning without pre-determined rules, alternative would be to design a loss function where matching the melody in training data over n consecutive notes of threshold is given penalty. Designing a more appropriate loss function remains as our future work. On the other hand, generating songs with parameters obtained at different stages within the 'safe zone' of training leads to diversity of melodies, even when the input vectors are identical. This property nicely complements our relatively low-dimensional input representation.

5 Conclusion & Future Works

In this paper, we proposed a novel model to generate melody for pop music. We generate melody with word representation of notes and their properties, instead of training multiple layers for each property, thereby reducing the complexity of learning. We also proposed a regularization model to control the outcome. Finally, we implemented part-dependent melody generation which helps the generated song preserve the overall structure, along with a publicly available dataset. Experimental results demonstrate that our model can generate songs whose melody sounds more like human-written ones, and is more well-structured than previous models. Moreover, people found it more difficult to distinguish the songs from our model from human-written songs than songs from previous models. On the other hand, examining other styles such as music of minor scale, or incorporating further properties of notes, such as intensity or vibrato, has not been examined yet, and remains as future work. As discussed in Section 4, learning to model the correlations among different instruments also remains to be done, and designing an appropriate loss function for the task is one of the most critical tasks to be done. We plan to constantly update our dataset and repository, addressing the future works.

REFERENCES

Hang Chu, Raquel Urtasun, and Sanja Fidler. Song from PI: A musically plausible network for pop music generation. In *ICLR Workshop*, 2017.

Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. In *ICLR Workshop*, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

Gaëtan Hadjeres and François Pachet. Deepbach: a steerable model for bach chorales generation. In *ICML*, 2017.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. In *ISMIR*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Bruce L. Jacob. Algorithmic composition as a model of creativity. *Organised Sound*, 1(3), 1996.

Natasha Jaques, Shixiang Gu, Richard E. Turner, and Douglas Eck. Tuning recurrent neural networks with reinforcement learning. In *ICLR Workshop*, 2017.

Andrej Karpathy and Fei-Fei Li. Deep Visual-Semantic Alignments for Generating Image Descriptions. In *CVPR*, 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

D.J. Levitin. *This is Your Brain on Music: The Science of a Human Obsession*. Dutton, 2006.

George Papadopoulos and Geraint Wiggins. Ai methods for algorithmic composition: A survey, a critical view and future prospects. In *AISB Symphosium on Musical Creativy*, 1999.

Peter Potash, Alexey Romanov, and Anna Rumshisky. Ghostwriter: Using an LSTM for automatic rap lyric generation. In *EMNLP*, 2015.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

Scott Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. Learning what and where to draw. In *NIPS*, 2016.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1), 2014.

A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236), 1950.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and Tell: A Neural Image Caption Generator. In *CVPR*, 2015.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Richard Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *ICML*, 2015.