Journal of Korea Multimedia Society Vol. 21, No. 8, August 2018(pp. 849-857)
https://doi.org/10.9717/kmms.2018.21.8.849

RNN Auto-Encoder의 시계열 임베딩을 이용한 자동작곡

김경환†, 정성훈††

Automatic Composition using Time Series Embedding of RNN Auto-Encoder

Kyung Hwan Kim†, Sung Hoon Jung††

ABSTRACT

In this paper, we propose an automatic composition method using time series embedding of RNN Auto-Encoder. RNN Auto-Encoder can learn existing songs and can compose new songs from the trained RNN decoder. If one song is fully trained in the RNN Auto-Encoder, the song is embedded into the vector values of RNN nodes in the Auto-Encoder. If we train a lot of songs and apply a specific vector to the decoder of Auto-Encoder, then we can obtain a new song that combines the features of trained multiple songs according to the given vector. From extensive experiments we could find that our method worked well and generated various songs by selecting of the composition vectors.

Key words: Automatic Composition, RNN, Auto-Encoder

1. 서 론

최근 인공지능 기술이 발전됨에 따라 대부분의 산업분야에 인공지능 기술을 접목하여 다양한 가치를 만들어내고 있다. 그러한 것으로 인공지능 기술을 이용한 의료정보 처리, 음성 인식을 이용한 인공지능 스피커, 자연어 처리 기술을 이용한 인공지능 챗봇, 영상 인식 기술을 이용한 인공지능 카메라 어플리케이션 등이 있다. 더 나아가 그 동안 인간의 고유 영역이라고 여겨지는 문화 예술 분야에서도 다양한 연구가 진행되고 있다. 인공지능이 그린 그림이나 소설, 시와 같은 창작물들이 전문가들에게 인간수준에 근접했다고 종종 평가된다.

창작분야 중 작곡 분야는 비교적 오래전부터 알고리즘 작곡이라는 이름으로 존재해왔으며 진화알고리즘이나 인공신경망 등의 인공지능 기법도 사용해왔다 [1-8]. 그러나 오랜 연구에도 불구하고 작곡가 수준의 곡을 작곡하지는 못하였다 [9,10]. 우리는 작곡가 수준의 작곡을 위하여 비교적 최근에 작곡을 자동으로 하는 연구를 시작하였으며 주로 인공신경망을 이용해서 연구를 진행해왔다 [9-13]. 인간 작곡가가 작곡할 때 기존 곡을 듣고 영감을 받아 새로운 곡을 작곡하듯이 인공신경망에 기존 곡을 학습시킨 후 학습된 신경망을 이용해 새로운 곡을 작곡하게 하였다. 하지만 인공신경망이 만들어 낸 곡은 시계열 데이터로 변환한 멜로디 데이터를 학습한 것이기 때

※ Corresponding Author : Sung Hoon Jung, Address: (02876) 116 Samseongyoro-16Gil Seongbuk-gu, Seoul, Korea, TEL : +82-2-760-4344, FAX : +82-2-760-4435, E-mail : shjung@hansung.ac.kr
Receipt date : Jul. 10, 2018, Approval date : Jul. 24, 2018
† Dept. of Electronics and Information Eng., Hansung University (E-mail : khsyee@gmail.com)
†† Dept. of Electronics and Information Eng., Hansung University
※ This research was financially supported by Hansung University. The preliminary results of this research were presented at the 2018 Summer Conference of the Digital Contents Society of Korea.

문에 박자, 화성 그리고 조성 측면에서 음악적 이론과는 맞지 않는 곡을 출력하였다. 이를 보완하기 위해 박자, 화성, 그리고 조성 후처리 알고리즘을 출력된 곡에 적용하여 음악적 이론을 만족하는 곡을 출력시켰다[10-12].

기존 연구에서는 곡을 학습하는 인공신경망으로 80년대에 개발된 전 방향 인공신경망(Feedforward Artificial Neural Network)을 사용하였다. 일반적으로 전 방향 인공신경망은 시계열데이터를 학습하지 못하기 때문에 학습할 곡의 데이터를 회귀적으로 구성하여 학습하였다[9,10]. 그러나 이렇게 회귀적으로 학습데이터를 구성하여 전 방향 인공신경망에 학습을 하는 것은 학습할 곡의 개수가 많지 않을 때는 큰 문제가 없지만 학습할 데이터가 많은 경우 각각의 곡이 적절히 임베딩 되지 못하는 문제가 있다.

이러한 문제를 해결하고자 본 논문에서는 다수의 곡을 잘 학습하기 위한 새로운 신경망 구조를 제안한다. 우리는 시계열 데이터를 학습하기에 적합한 Recurrent Neural Network(RNN)과 학습 데이터의 특징을 은닉층에 압축시키는 신경망 구조인 Auto-Encoder를 결합했다. 두 가지를 결합한 RNN Auto-Encoder는 시계열을 학습할 수 있기 때문에 데이터를 회귀적으로 구성하지 않아도 멜로디 데이터가 잘 학습된다. 또한 다수의 곡을 학습할 때 각 곡 데이터의 특성이 압축된 공간에 저장되는 효과가 있기 때문에 학습된 신경망으로 새로운 곡을 작곡할 때 신경망의 입력에 따라 다양한 특성을 갖는 곡으로 작곡할 수 있다.

우리는 제안한 RNN Auto-Encoder를 텐서플로우 상에서 구현하였다. Auto-Encoder상의 Encoder 와 Decoder는 128개의 은닉노드를 갖는 RNN으로 구성된다. 긴 곡도 잘 학습할 수 있도록 RNN은 LSTM (Long Short-Term Memory) 모델을 사용하였다. 학습할 곡은 멜로디와 박자로 나뉘어져서 one-hot encoding 된 형태로 두 개의 RNN Auto-Encoder에 학습된다. 하나의 곡 전체가 Auto-Encoder의 Encoder에 입력된 후 Encoder에 저장된 128개의 상태가 Decoder에 복사되고 시작태그인 <S>와 함께 Decoder에 학습된다. Decoder의 출력은 완전연결계층(Fully Connected Layer)을 거쳐서 출력되며 원래와 동일한 시계열이 생성되도록 반복적으로 학습된다.

학습이 종료되면 Auto-Encoder의 Decoder를 이용해 새로운 곡을 작곡한다. Decoder에 시작 태그와 함께 새로운 128개의 벡터를 넣어 주면 새로운 곡이 출력된다. 만약 학습된 하나의 곡의 벡터와 동일한 것을 넣어주면 학습된 곡과 동일한 곡이 출력된다. 그러므로 많은 곡을 학습시킨 후 작곡을 위한 벡터 입력을 적절히 넣어주면 다수의 학습시킨 곡의 특징을 갖는 곡이 출력된다. 본 논문에서는 작곡을 위한 새로운 벡터를 넣는 방법으로 학습된 두 곡의 벡터 평균을 넣는 방법과 무작위 벡터를 넣는 방법에 대하여 실험하였다. 실험결과 두 곡의 벡터 평균을 넣는 경우 해당 곡의 특징이 잘 섞이는 것을 볼 수 있었으며 무작위 벡터를 넣는 경우 다양한 곡의 특징이 융합되는 것을 볼 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서 기존에 연구한 자동작곡 방법을 소개한다. 3장에서 본 논문에서 제안하는 RNN Auto-Encoder와 이를 이용해 자동 작곡하는 방법을 기술한다. 4장에서 실험결과에 대해 설명하며 5장의 결론으로 마무리한다.

2. 기존 자동작곡 방법

우리는 기존연구에서 곡의 데이터를 회귀적으로 구성하여 이를 전방향 인공신경망에 오류역전파알고리즘으로 학습하였다[9-13]. 학습에 사용한 인공신경망은 내부적으로 세 개로 구성하여 각각 음표/쉼표/박자를 학습하였다. 학습할 곡의 멜로디는 음표/쉼표/박자의 시계열 숫자 데이터로 표현하여 학습데이터로 사용하였다. 음표는 전체 7옥타브 중에 2~4옥타브를 사용하여 1~36까지의 숫자로 표현하였다. 쉼표의 경우는 음표와는 의미가 다르기 때문에 음표에서 사용하지 않는 숫자로 표현할 필요가 있다. 그래서 우리는 쉼표의 경우 숫자 50을 사용하여 시계열 데이터를 만들었다[9,10]. 박자는 4/4박자에서 16분 음표를 1로 가정하고 하나의 점음표까지만 허용하여 1, 2, 3, 4, 6, 8, 12, 16의 총 8개의 박자를 사용하였다[9].

학습할 곡이 음표/쉼표/박자의 시계열 데이터로 변환된 후에는 전방향 인공신경망에 학습하기 위하여 학습데이터를 회귀적으로 구성한다[9]. 회귀적 학습데이터를 구성하는 방법은 인공신경망의 입력을 10개로 사용하는 경우 학습 데이터 10개를 입력으로 넣고 11번째 데이터를 출력하는 방법으로 구성한다.

그 다음 학습데이터는 첫 번째 데이터를 제거하고 두 번째 데이터로부터 11번째 데이터를 입력으로 하여 12번째 데이터를 출력하는 방식으로 모든 곡의 데이터를 구성한다[9]. 그러나 이렇게 구성을 하면 대부분의 곡이 반복적으로 구성되기 때문에 입력은 같은데 출력은 다르게 학습데이터가 만들어져서 인공신경망이 학습이 안 되는 문제가 발생한다[10]. 예를 들어 10개의 입력은 같은데 11번째의 출력이 다른 두 개의 학습 데이터가 있다면 인공신경망은 두 학습 데이터를 구분해서 학습할 수 없다. 이를 해결하기 위하여 입력 데이터에 마디구분을 추가하여 학습 데이터를 만들어서 학습시켰다[9,10]. 그러나 이러한 문제는 본 연구에서처럼 RNN을 사용할 경우에는 RNN이 내부에 상태를 갖기 때문에 발생하지 않는다.

마디구분은 음표/쉼표/박자별로 해당 데이터와 같은 범위의 무작위 수를 생성하여 만든다. 마디구분으로 사용하는 수는 그 값 자체가 중요한 것이 아니라 학습 시와 작곡 시에 마디를 구분만하면 되는 것이기 때문에 무작위로 만들어도 된다. 학습된 인공신경망으로 작곡 할 때 학습 시 사용한 마디구분을 이용하여 곡의 위치를 설정할 수 있으며 또한 학습 시에 사용한 마디구분 값을 변형함으로서 작곡하는 멜로디에 변형을 가할 수 있는 등 많은 장점이 있다. 더 나아가 작곡 시에 마디구분을 반복적으로 배치함으로서 반복적인 곡을 생성할 수 있다. 보다 자세한 것은 기존 논문[9,10]을 참고하기 바란다.

학습 후에는 학습한 곡과 다른 초기 음표/쉼표/박자를 만들어 넣어주면 나머지 음표/쉼표/박자를 만들어준다. 이러한 초기 음표/쉼표/박자는 해당 시계열 데이터에서 발생할 수 있는 범위에서 무작위로 만들어도 되며 이 초기 음표/쉼표/박자에 따라서 다양한 다른 곡이 출력된다[11,12]. 그러나 인공신경망이 출력한 음표/쉼표/박자는 학습된 음표/쉼표/박자 공간의 값을 출력한 것으로 음악적 이론에 맞지 않는 경우가 종종 발생한다. 박자의 경우 못갖춘마디가 발생하고 음표의 경우 특정 조성에 없는 음이 발생하거나 마디 내에서 어울리지 않는 화성의 음표가 발생하기도 한다. 이를 해결하기 위하여 박자후처리, 화성후처리, 조성후처리를 수행하였다[11]. 또한 음악적으로 완성된 곡을 위해서는 곡이 구성을 갖출 필요가 있다. 인공신경망이 작곡한 곡은 구성을 갖추지 못한 경우가 많았다. 구성을 갖춘 곡의 생성을 위해 신경망을 계층적으로 구성하는 방법을 고안했다[13]. 계층적 인공신경망으로 학습 후 작곡한 곡은 전, 중, 후의 일정한 구성을 갖는 곡으로 생성되었다.

3. RNN Auto-Encoder를 이용한 자동작곡

본 절에서는 본 논문에서 제안한 RNN Auto-Encoder 의 구조 및 학습방법 그리고 작곡방법을 설명한다. 먼저 RNN Auto-Encoder와 관련이 있는 RNN 인공신경망과 Auto-Encoder를 설명하고 RNN Auto-Encoder의 구조 및 학습방법을 설명한다. 마지막으로 RNN Auto-Encoder를 이용하여 다수의 곡을 학습하고 작곡하는 방법을 설명한다.

3.1 Recurrent Neural Network(RNN)

RNN은 신경망의 한 종류로 은닉층이 순환하는 구조를 가진 신경망이다. 순환하는 구조를 가졌기 때문에 연속성을 갖는 시계열데이터를 학습하기에 적합하다. 그러나 일반적인 RNN의 경우 단기적인 의존관계만 학습할 수 있기 때문에 장기적인 의존관계를 학습하는 것은 부적합하다. 본 논문에서는 RNN의 종류 중 장기의존성 문제를 해결한 LSTM(Long Short Term Memory) 구조를 사용한다. Fig. 1의 (a)와 (b)는 일반 RNN과 LSTM의 구조의 차이를 보여준다. 일반 RNN의 경우 은닉층의 상태가 단순히 현재 입력(x_t)과 이전 은닉층의 상태(h_{t-1})로 결정된다.

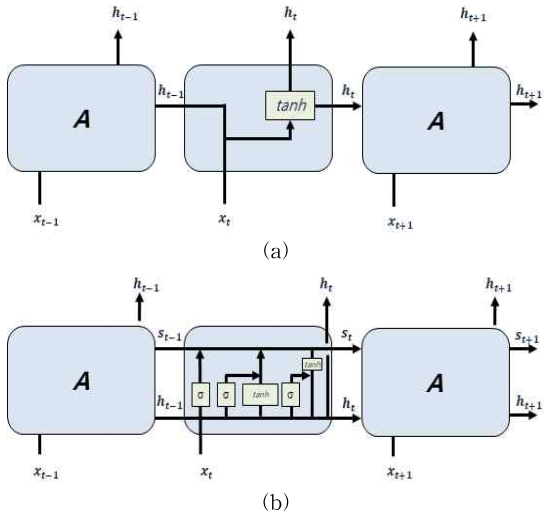

Fig. 1. Structure of RNN and LSTM. (a) RNN (b) LSTM.

그러나 이렇게 하면 시간 간격이 긴 정보, 즉 오래 전에 입력한 정보에 의존해서 출력을 내야하는 경우 오래 전에 입력된 정보가 거의 사라져 이러한 의존 관계를 학습할 수 없다. 이를 극복하기 위하여 LSTM에서는 은닉층의 상태정보(h_t)와 함께 RNN의 셀 상태 정보(s_t)를 추가하였으며 셀 상태 정보도 은닉층의 상태정보처럼 순환하도록 하였다. 그리고 장기의 존성을 해결하기 위하여 셀 상태정보의 순환에 이전 셀 상태정보(s_{t-1})를 잊기 위한 망각게이트(forget gate)와 현재 입력정보를 기억하기 위한 입력게이트(input gate)를 추가하였다. 망각 게이트와 입력게이트는 현재입력(x_t)과 은닉층의 이전 상태정보(h_{t-1})를 이용하여 만들어진다. 결국 일반 RNN에서는 현재입력(x_t)과 은닉층의 이전 상태정보(h_{t-1})로 은닉층의 현재 상태정보(h_t)를 만드는 반면 LSTM에서는 현재 입력(x_t)과 은닉층의 이전 상태정보(h_{t-1})로 은닉층의 임시 상태정보와 망각게이트, 입력게이트, 출력게이트를 만든다. 셀의 현재상태(s_t)는 셀의 이전 상태(s_{t-1})가 망각게이트를 통과한 값과 은닉층의 임시상태가 입력게이트를 통과한 값으로 결정된다. 은닉층의 현재 상태(h_t)는 셀의 현재 상태(s_t)가 출력게이트를 통과한 값으로 결정된다.

3.2 Auto-Encoder

Auto-Encoder는 출력레이블 없이 입력 데이터만으로 신경망을 학습하여 은닉층에 데이터의 특징을 압축시키는 비지도 학습 방법이다. Encoder와 Decoder로 구성되어 있으며 Fig. 2에서 보듯이 서로가 반전된 구조를 가지고 있다. 높은 차원의 입력데이터를 낮은 차원의 코드로 만들기 때문에 코드에는 입력 데이터의 주요한 특징이 포함된다. 다수의 데이터를 Auto-Encoder에 학습하면 압축된 공간에 다수 데이터의 특징들이 매핑되는 효과가 있으며 Decoder에 임의의 코드를 넣어주면 해당 코드와 유사한 코드를 만드는 학습데이터들과 유사한 새로운 출력을 만들어 낼 수 있다.

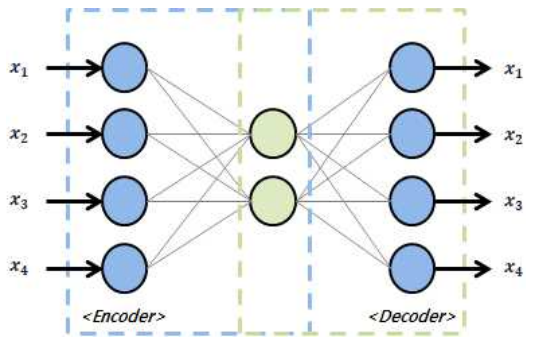

Fig. 2. Structure of Auto-Encoder.

3.3 RNN Auto-Encoder

본 논문에서는 RNN과 Auto-Encoder를 결합한 새로운 구조의 신경망을 제안한다. RNN Auto-Encoder는 기존 Auto-Encoder의 Encoder와 Decoder를 RNN으로 구성한다. Fig. 3은 본 논문에서 제안한 Auto-Encoder의 구조를 보여준다. Fig. 3(a)는 학습시의 구조를 보여주며 Fig. 3(b)는 작곡시의 구조를 보여준다.

RNN Auto-Encoder의 RNN Encoder는 one-hot encoding 된 시계열데이터가 입력되어 학습되며 RNN Decoder는 시작태그 <S>와 함께 입력 시계열데이터의 역순으로 출력되게 학습된다. RNN Decoder 의 출력을 입력 시계열데이터의 역순으로 학습시키는 이유는 RNN Encoder에서 모든 입력 시계열데이터

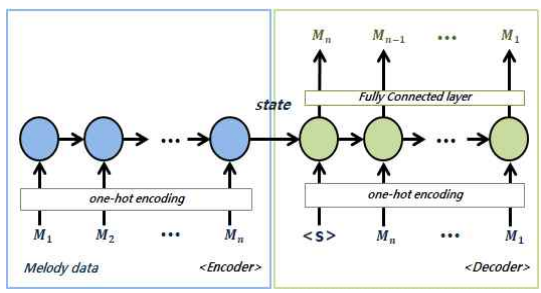

Fig. 3. Structure of RNN Auto-Encoder. (a) When training, the structure of RNN Auto-Encoder (b) When generating songs, the structure of RNN

가 입력된 후에 결정된 은닉층의 상태가 RNN Decoder의 초기 상태로 복사되기 때문이다. 결과적으로 Encoder와 Decoder가 대칭의 형태를 이루게 된다. 이러한 구조가 RNN Auto-Encoder의 학습에 이점을 갖고 있다[14]. RNN Decoder의 출력에 완전연결 층이 있는 이유는 one-hot encoding된 출력을 원래의 시계열데이터로 변환하려는 의도이다. 하나의 시계열 데이터가 학습되면 해당 곡은 하나의 RNN Encoder 은닉층 상태벡터로 수렴한다. 다수의 시계열 데이터를 학습하면 각각의 데이터는 각각의 은닉층 상태벡터로 수렴한다.

3.4 RNN Auto-Encoder를 이용한 작곡

Fig. 4는 RNN Auto-Encoder를 이용하여 작곡하는 과정을 보여준다. RNN Auto-Encoder를 이용한 자동작곡의 전체적인 과정은 Fig. 4의 (a)와 같다. 먼저 학습할 다수의 midi 파일을 전 처리 한다. 전처리는 화음이 있는 곡에 대해서 근음만을 추출하고 추출한 근음에 대해서 음표와 쉼표 그리고 박자를 추출한다. 음의 경우 학습 곡 데이터에 사용된 음의 범위를

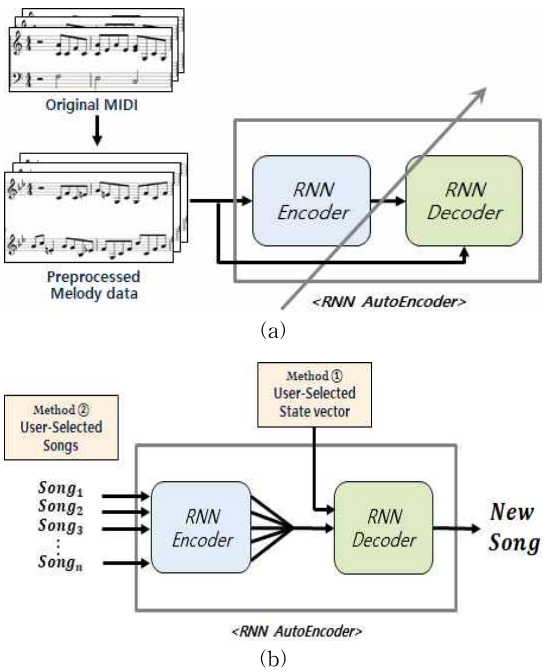

Fig. 4. Automatic Composition Process using RNN Auto-Encoder (a) Training process of RNN Auto-Encoder using MIDI files (b) Two ways to create a new song using the learned RNN Auto-Encoder.

고려해 2 옥타브 솔부터 6 옥타브 도(G2~C#6)까지 총 42개의 음을 사용한다. 박자의 경우 4분 음표를 1.0이라 할 때 2개의 점음표까지 허용하여 4.0, 3.5, 3.0, 2.0, 1.75, 1.5, 1.0, 0.75, 0.5, 0.25 총 10가지 박자를 사용한다. 추출한 정보는 one-hot encoding으로 변환되어 Auto-Encode의 학습데이터가 된다. 음표와 쉼표는 같이 one-hot encoding 되고 박자는 따로 one-hot encoding 된다. 그러므로 두 개의 Auto-Encoder에 음표와 쉼표 그리고 박자가 각각 학습된다. 화음을 제거하고 근음만으로 학습하는 이유는 화음까지 고려하면 one-hot encoding 될 정보가 너무 많아지고 그렇게 되면 입력데이터가 많아져 학습이 오래 걸리는 문제가 있다. 우리는 일단 근음만 학습시켜서 근음만 작곡하고 따로 화음을 처리하는 인공신경망을 개발할 예정이다.

Auto-Encoder의 학습은 One-hot encoding 된 시계열데이터를 RNN Encoder의 입력으로 하고 동일한 데이터를 RNN Decoder의 출력으로 하여 학습한다. 같은 방법으로 다수의 곡을 학습할 수 있으며 학습이 종료된 후에 작곡을 할 수 있다. 작곡하는 방법은 Fig. 4(b)의 방법 1에 있는 것처럼 시작 태그 <S>와 사용자가 선택한 상태벡터를 RNN Decoder에 넣어주면 된다. RNN Decoder는 시작태그와 상태벡터를 받아서 새로운 곡을 작곡한다. 작곡용 상태벡터는 사용자가 임의로 설정할 수도 있으나 기존 곡의 상태벡터를 이용하여 구할 수도 있다. Fig. 4(b)의 두 번째 방법에서 보여 주듯이 학습한 곡을 다수 개 선택하고 해당 곡의 RNN Encoder 출력 상태를 이용하여 만들 수 있다. 예를 들어 n개의 곡을 선택한 경우 해당 곡의 출력상태의 평균을 구해서 작곡용 상태벡터로 만들 수 있다. 이 경우 선택된 곡들의 특징이 작곡한 결과 곡에 반영된다. Fig. 5는 4개의 곡의 평균 상태벡터로 작곡 할 경우의 상황을 보여준다. 곡을 대량으로 학습할수록 점점 더 상태 공간에 매핑되는 곡이 많아지며 그러므로 점점 더 다양한 곡의 특징이 반영된 곡이 출력될 수 있다. 출력된 곡을 midi 파일로 만들면 작곡이 완료된다.

4. 실험 결과

우리는 제안한 RNN Auto-Encoder의 성능을 보기 위하여 텐서플로우 상에서 구현하고 다양한 실험을 수행하였다. RNN Auto-Encoder의 RNN Encod-

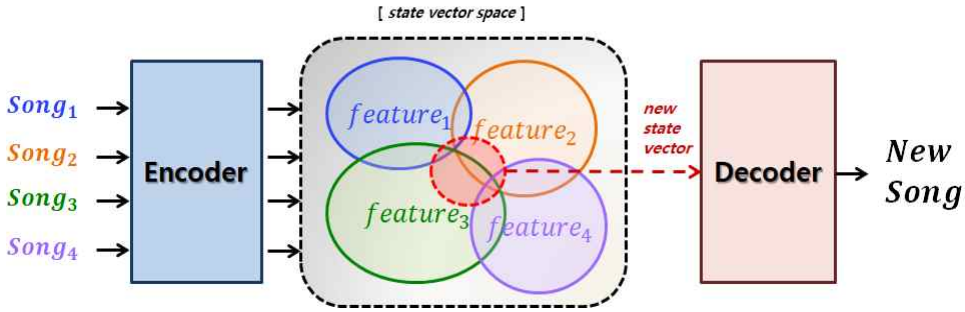

Fig. 5. State vector space of trained RNN Auto-Encoder.

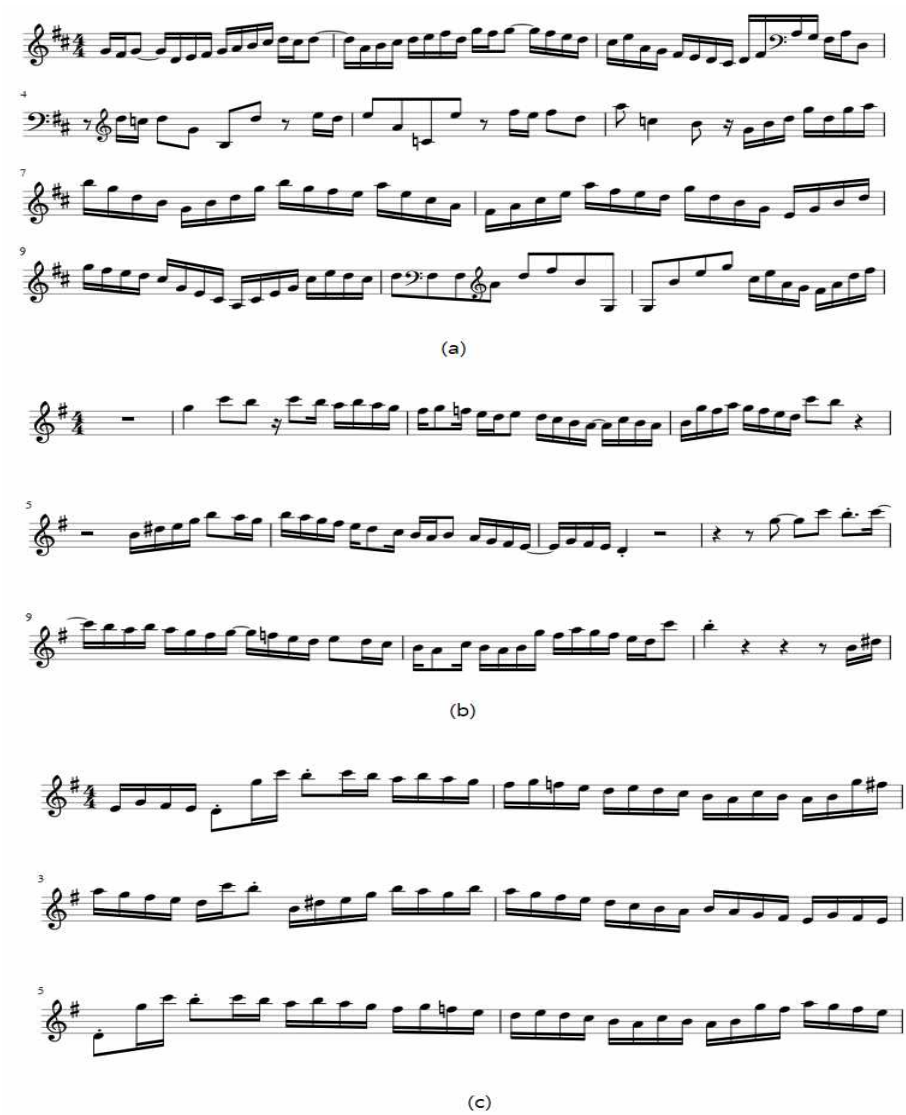

Fig. 6. Generated song (a) first training song, (b) second training song, (c) generated song using average state vector of two songs.

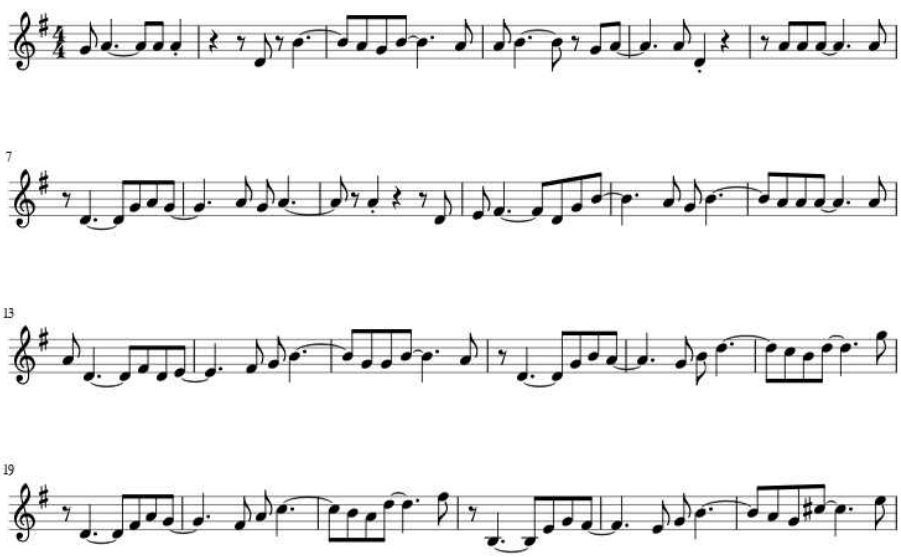

Fig. 7. Generated song using a random vector after training 26 number of songs.

er와 RNN Decoder는 단층의 128개의 은닉노드를 갖는 구조를 사용하였다. 하나의 곡을 학습한 후 해당 곡의 상태벡터를 넣어주면 원곡이 그대로 출력됨을 볼 수 있었다. 다수의 곡을 학습한 후 학습한 곡 중 두 곡의 평균상태벡터를 넣어주면 두 곡의 특징이 적절히 섞인 곡이 출력됨을 볼 수 있었다.

Fig. 6은 그 중 하나로 (a)와 (b)는 학습한 곡 중 두 곡의 일부를 보여주며 (c)는 (a) 와 (b) 두 곡의 상태벡터의 평균으로 작곡한 결과의 일부를 보여준다. Fig. 6에서 보듯이 작곡된 곡이 학습된 두 곡의 특징을 모두 반영한 것을 볼 수 있다. 대규모 학습 실험으로는 26개의 곡을 학습하고 임의의 상태벡터를 넣어서 작곡한 결과를 살펴보았다. Fig. 7은 하나의 결과를 보여준다. 이 경우에도 학습한 곡들의 특징이 작곡한 곡에 나타났으며 임의로 넣어준 벡터공간에 학습된 곡들의 특징이 나타남을 알 수 있었다. 다만 학습한 곡들이 매핑되는 상태공간과 작곡용으로 넣어준 상태벡터 공간 사이의 관계를 정확히 분석할 수는 없기 때문에 작곡된 곡이 어떤 곡들의 특징을 주로 반영했는지 분석하는 것은 매우 어려웠다. 그러나 작곡이라는 것은 기존 곡들의 특징을 적절히 융합하는 것이 중요하기 때문에 큰 문제는 되지 않는다. 장르별로 다수의 곡을 학습하고 장르별 평균과 편차를 이용하여 작곡용 상태벡터를 생성하면 장르별 작곡도 가능할 수 있을 것으로 판단된다. 차후 연구에서 장르별 대량의 곡을 학습하여 장르별로 작곡하는 것을 실험해볼 예정이다.

Fig. 6의 작곡결과와 Fig. 7의 작곡결과는 midi 파일로 아래의 사이트를 접속하여 들어볼 수 있다.

- http://itsys.hansung.ac.kr/dn/make_test_20180702-035219_988-v01_988-v02.mid
- http://itsys.hansung.ac.kr/dn/make_test_20180702-035219_randn_input4.mid

5. 결 론

본 논문에서는 RNN Auto-Encoder를 이용한 자동작곡 방법을 제안하였다. RNN Auto-Encoder를 구현하여 곡을 학습시켰다. RNN Auto-Encoder에 곡을 학습하면 동일한 곡을 출력하는 상태벡터로 해당 곡이 임베딩 되었다. 다수의 곡을 학습한 후 학습한 곡 중 두 곡의 상태벡터 평균을 RNN Decoder에 넣으면 두 곡의 특징이 나타나는 곡을 출력하였다. 또한 다수의 곡을 학습한 후에 임의의 상태벡터를 RNN Decoder에 넣어주면 기존 곡들의 상태벡터와 작곡을 위해 넣어준 상태벡터의 관계를 통하여 기존의 학습된 곡들의 특징을 갖는 새로운 곡을 출력할 수 있었다. 본 연구에서는 근음만을 학습하고 근음만을 작곡하는 것을 구현하였다. 추후 작곡된 근음으로 화성을 만들어 화성이 있는 곡을 만드는 연구를 진행

할 예정이다.

REFERENCE

[1] B. Johanson and R. Poli, "GP-music: An Interactive Genetic Programming System for Music Generation with Automated Fitness Raters," *Proceeding of the Third Annual Conference*, pp. 181-186, 1998.

[2] N. Tokui and H. Iba, "Music Composition with Interactive Evolutionary Computation," *Proceeding of the Third International Conference on Generative Art*, pp. 215-226, 2000.

[3] C. Chen and R. Miikkulainen, "Creating Melodies with Evolving Recurrent Neural Networks," *Proceedings of the 2001 International Joint Conference on Neural Networks*, pp. 2241-2246, 2001.

[4] T. Oliwa and M. Wagner, "Composing Music with Neural Networks and Probabilistic Finite-state Machines," *Applications of Evolutionary Computing*, pp. 503-508, 2008.

[5] H. Kim, B. Kim, and B. Zhang, "Learning Music and Generation of Crossover Music Using Evolutionary Hypernetworks," *Proceeding of Korea Computer Congress*, pp. 134-138, 2009.

[6] G. Bickerman, S. Bosley, P. Swire, and Rober M. Keller, "Learning to Create Jazz Melodies Using Deep Belief Nets," *Proceeding of the International Conference on Computational Creativity*, pp. 228-237, 2010.

[7] A.E. Coca, R.A.F. Romero, and L. Zhao, "Generation of Composed Musical Structures Through Recurrent Neural Networks Based on Chaotic Inspiration," *Proceeding of International Joint Conference on Neural Networks*, pp. 3220-3226, 2011.

[8] J.D. Fernandez and F. Vico, "AI Methods in Algorithmic Composition: A Comprehensive Survey," *Journal of Artificial Intelligence Research*, Vol. 48, pp. 513-582, 2013.

[9] J. Cho, E.M. Ryu, J. Oh, and S.H. Jung, "Training Method of Artificial Neural Networks for Implementation of Automatic Composition Systems," *Korea Information Processing Society Transactions on Software and Data Engineering*, Vol. 3, No. 8, pp. 315-320, 2014.

[10] J. Oh, J. Song, K. Kim, and S.H. Jung, "Automatic Composition Using Training Capability of Artificial Neural Networks and Chord Progression," *Journal of Korea Multimedia Society*, Vol. 18, No. 11, pp. 1358-1366, 2015.

[11] K. Kim and S.H. Jung, "Postprocessing for Tonality and Repeatability, and Average Neural Networks for Training Multiple Songs in Automatic Composition," *Journal of Korean Institute of Intelligent Systems*, Vol. 26, No. 6, pp. 445-451, 2016.

[12] K. Kim and S.H. Jung, "Adoption of Artificial Neural Network for Rest, Enhanced Postprocessing of Beats and Initial Melody Processing for Automatic Composition System," *Journal of Korea Digital Contents Society*, Vol. 17, No. 6, pp. 449-459, 2016.

[13] K. Kim and S.H. Jung, "Automatic Generation of a Configured Song with Hierarchical Artificial Neural Networks," *Journal of Korea Digital Contents Society*, Vol. 18, No. 4, pp. 641-647, 2017.

[14] S. Nitish, M. Elman, and S. Ruslan, "Unsupervised Learning of Video Representations Using LSTMs," *Proceeding of the 32nd International Conference on Machine Learning*, Vol. 37, No. 10, pp. 843-852, 2015.

김 경 환

2018년 2월 한성대학교 전자정보공학과(공학사)
관심분야 : 인공지능, 융합공학, 임베디드 시스템

정 성 훈

1988년 2월 한양대학교 전자공학과 (공학사)
1991년 2월 KAIST 전기및전자공학과 (공학석사)
1995년 2월 KAIST 전기및전자공학과 (공학박사)

1996년~현재 한성대학교 전자정보공학과 교수
관심분야 : 인공지능, 융합공학, 시스템생물학