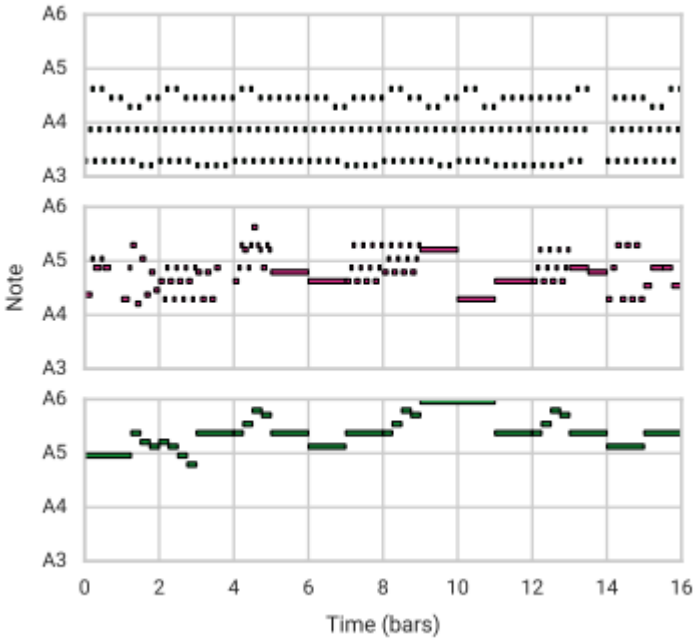
A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music

1. 소개

VAE(Variational Autoencoder)는 자연 데이터에 대해 의미적으로 의미 있는 잠재 표현을 생성하기 위한 효과적인 모델임이 입증되었지만 지금까지 순차적 데이터에 대한 적용이 제한적이며 기존의 **recurrent VAE** 모델은 **long-term structure**의 시퀀스를 모델링하는 데 어려움이 있습니다.

논문에서는 이 문제를 해결하기 위해 먼저 입력의 하위 시퀀스에 대한 임베딩을 출력한 다음 이러한 임베딩을 사용하여 각 하위 시퀀스를 독립적으로 생성하는 계층적 디코더의 사용을 제안하였습니다.

이 구조는 모델이 잠재 코드(Latent Code)를 활용하도록 하여 **recurrent VAE**의 문제로 남아 있는 **"posterior collapse"** 문제를 방지합니다. 우리는 이 아키텍처를 음표의 모델링 시퀀스에 적용하고 "flat" 베이스라인 모델보다 훨씬 더 나은 샘플링, 보간 및 재구성 성능을 나타내는것을 확인하였습니다.



MusicVAE를 사용한 잠재 공간 평균의 시연. 상단 및 하단 시퀀스에 대한 잠재 코드는 중간 시퀀스를 생성하기 위해 우리 모델에 의해 평균화되고 디코딩됩니다.

GAN과 같은 암시적 모델과 PixelCNN과 같은 명시적 심층 자기회귀 모델을 포함하여 심층 생성 모델링에 다양한 방법이 사용되었습니다. 이 작업에서는 VAE와 같은 깊은 잠재 변수 모델에 중점을 둡니다.

이러한 모델은 p(x|z) 와 p(z)를 명시적으로 모델링 할 수 있다는 장점을 가집니다. (z는 잠재 벡터) 잠재 벡터는 주어진 데이터 포인트의 관련 특성을 캡처하고 데이터 세트의 변동 요인을 분리합니다.

이러한 특징을 활용하여 응용프로그램의 창의적 활용이 가능한데 예를 들어, 주어진 속성을 소유한 모든 데이터 포인트에 대한 잠재 코드를 평균화하면 데이터 예를 목표로 변경하는 데 사용할 수 있는 소위 속성 벡터가 생성됩니다. 데이터 포인트를 일부 속성(예: 갈색 머리를 가진 사람의 사진)으로 인코딩하여 잠재 코드를 얻고 해당 속성 벡터("갈색 머리" 벡터)를 빼고 다른 속성 벡터("금발 머리")를 추가합니다. 결과 잠재 벡터를 디코딩하면 속성이 교환된 초기 지점의 사실적인 표현을 생성할 수 있습니다(금발 머리를 가진 동일한 사람). 또 다른 예로서, 궤적 상의 잠재 벡터와 디코딩 포인트 사이의 보간은 부드럽고 의미론적으로 의미 있는 방식으로 끝의 특성 사이를 모핑하는 현실적인 중간 데이터 포인트를 생성할 수 있습니다.

심층 잠재 변수 모델에 대한 대부분의 작업은 이미지와 같이 차원이 고정된 연속 값 데이터에 중점을 둡니다. 순차 데이터 모델링은 일반적이지 않으며, 특히 악보와 같은 개별 토큰 시퀀스는 일반적으로 **autoregressive decoder**를 사용해야 합니다. 이는 부분적으로 **autoregression**이 종종 **autoencoder**가 잠재 코드를 무시할 만큼 충분히 강력하기 때문입니다.

이 격차를 해결하기 위해 논문에서는 **hierarchical recurrent decoder**가 있는 새로운 **sequential auto encoder**를 소개합니다. 이는 recurrent VAE로 long-term 구조를 모델링하는 앞에서 언급한 문제를 극복하는 데 도움이 됩니다. "flat" 디코더 RNN을 사용하는 기준 모델보다 훨씬 더 긴 시퀀스를 효과적으로 자동 인코딩할 수 있음을 실험적으로 보여줍니다.

2. 모델

2.1. Recurrent VAE

MusicvAE 에서는 RNN 모델을 사용합니다. 구체적으로 인코더 qλ(z|x)는 RNN모델로 입력 시퀀스 x = {x1, x2, . . . , xT }를 받아 은닉 상태 h1, h2, . . . , hT를 생성합니다.. 그러면 잠재 코드 z에 대한 분포의 매개변수가 hT의 함수로 설정됩니다. 디코더 pθ(x|z)는 샘플링된 잠재 벡터 z를 사용하여 자기회귀적으로 출력 시퀀스 y = {y1, y2, . . . , YT }를 만드는디코더 RNN의 초기 상태를 설정합니다. 모델은 입력 시퀀스(즉, yi = xi , i ∈ {1, ... , T})를 재구성하고 표준 VAE에서와 같이 이전 p(z)에 가까운 대략적인 사후 qλ(z|x)를 학습하도록 훈련됩니다.

이 접근 방식에는 두 가지 주요 단점이 있습니다. 첫째, RNN은 일반적으로 그 자체로 시퀀스의 강력한 자기회귀 모델로 사용되기 때문에 **recurrent VAE**의 디코더 자체는 데이터의 효과적인 모델을 생성하기에 충분히 강력하며 잠재 코드를 무시할 수 있습니다. 둘째, 모델은 전체 시퀀스를 단일 잠재 벡터로 압축해야 합니다. 이 접근 방식은 짧은 시퀀스에서 작동하는 것으로 나타났지만 시퀀스 길이가 증가함에 따라 실패하기 시작합니다. 이러한 문제를 해결하기 위해 디코더에 **hierarchical RNN**을 사용하여 이러한 문제를 극복하는 **latent variable autoencoder**를 제시합니다.

2.2. Bidirectional Encoder

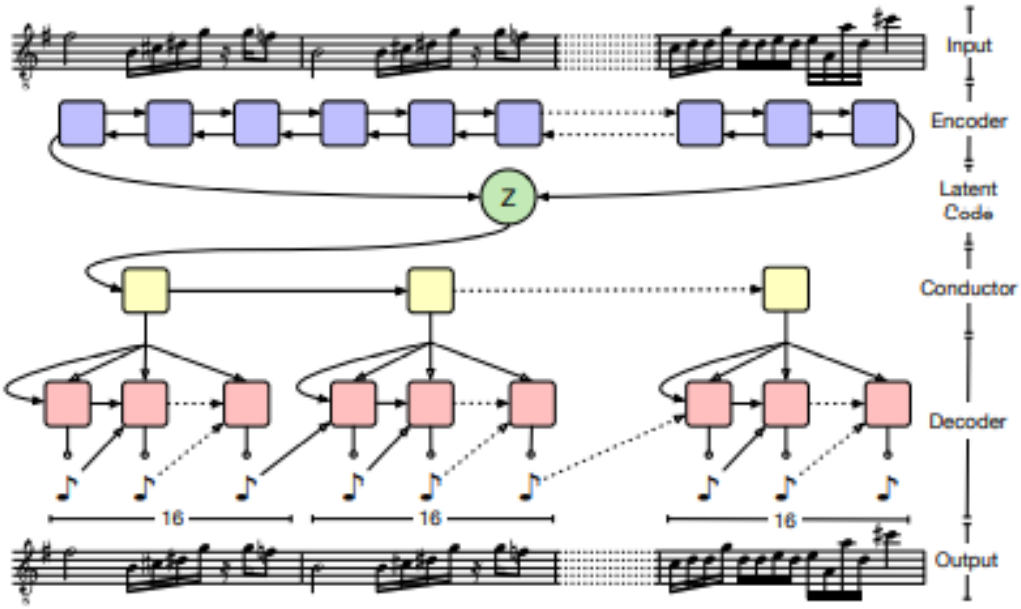


Figure 2. Schematic of our hierarchical recurrent Variational Autoencoder model, MusicVAE.

인코더 qλ(z|x)의 경우 2계층 양방향 LSTM 네트워크를 사용합니다. 입력 시퀀스 x = {x1, x2, . . . , xT }는 두 번째 양방향 LSTM 계층에서 최종 상태 벡터

$$\overrightarrow{h}_T, \overleftarrow{h}_T$$

를 얻습니다. 그런 다음 이들은 연결되어 hT를 생성하고 두 개의 **fully-connected layer**에 공급되어 잠재 분포 매개변수 μ 및 σ를 생성합니다.

$$\mu = W_{h\mu} h_T + b_\mu \quad (6)$$
$$\sigma = \log\left(\exp(W_{h\sigma} h_T + b_\sigma) + 1\right) \quad (7)$$

$$W_{h\mu}, W_{h\sigma} \text{ and } b_\mu, b_\sigma$$

는 각각 가중치 행렬과 bias 벡터입니다. 우리의 실험에서 우리는 모든 레이어와 512개의 잠재 차원에 대해 2048의 LSTM 상태 크기를 사용합니다. VAE의 표준과 같이 μ와 σ는 잠재 분포를 매개변수화합니다. 양방향 순환 인코더를 사용하면 입력 시퀀스에 대한 longer-term context에 대한 잠재적인 분포의 매개변수화를 이상적으로 사용할 수 있습니다.

2.3. Hierarchical Decoder

2.1.에서 언급한 단순 스택 RNN이 가지는 문제점을 해결하기 위하여 디코더에 새로운 **Hierarchical RNN**을 사용하였습니다. 입력 시퀀스 x가 끝점 iu가 있는 U개의 겹치지 않는 부분 시퀀스 yu로 분할될 수 있다고 가정합니다.

$$y_u = \{x_{i_u}, x_{i_u+1}, x_{i_u+2}, \ldots, x_{i_{u+1}-1}\} \quad (8)$$
$$\rightarrow \mathbf{x} = \{y_1, y_2, \ldots, y_U\} \quad (9)$$

여기서 iU+1 = T의 특수한 경우를 정의합니다. 그런 다음 잠재 벡터 z는 **fully-connected layer**를 통과한 다음 tanh 활성화를 거쳐 **"conductor" RNN**의 초기 상태를 얻습니다. conductor RNN은 각 하위 시퀀스에 대해 하나씩 U 임베딩 벡터 c = {c1, c2, . . . , cU }를 제공합니다. 실험에서 은닉 상태 크기가 1024 및 512 출력 차원인 **conductor**에 대해 two-layer 단방향 LSTM을 사용합니다.

컨덕터가 일련의 임베딩 벡터 c를 생성하면 각각은 공유된 완전 연결 레이어를 개별적으로 통과한 다음 tanh 활성화를 거쳐 최종 하위 레이어 디코더 RNN의 초기 상태를 생성합니다. 그런 다음 디코더 RNN은 softmax 출력 레이어를 통해 각 하위 시퀀스 yu에 대한 출력 토큰에 대한 분포 시퀀스를 자동 회귀적으로 생성합니다. 최하위 디코더의 각 단계에서 현재 **conductor** 임베딩 cu는 입력으로 사용할 이전 출력 토큰과 연결됩니다. 논문에서는 디코더 RNN을 위해 레이어당 1024 유닛을 가진 2레이어 LSTM을 사용했습니다.

Conductor가 일련의 임베딩 벡터 c를 생성하면 각각은 공유된 **fully-connected layer**를 개별적으로 통과한 다음 tanh 활성화를 거쳐 최종 하위 레이어 디코더 RNN의 초기 상태를 생성합니다. 그런 다음 디코더 RNN은 softmax 출력 레이어를 통해 각 하위 시퀀스 yu에 대한 출력 토큰에 대한 분포 시퀀스를 자동 회귀적으로 생성합니다. 최하위 디코더의 각 단계에서 현재 **conductor** 임베딩 cu는 입력으로 사용할 이전 출력 토큰과 연결됩니다. 사용된 모델에서는 디코더 RNN을 위해 레이어당 1024 유닛을 가진 2레이어 LSTM을 사용했습니다.

3. 구현

3.1. 데이터 전처리

데이터는 Groove MIDI Dataset: https://magenta.tensorflow.org/datasets/groove 에서 제공하는 MIDI 파일을 사용하였습니다.

먼저 **magenta**의 **convert_directory** 함수를 활용하여 MIDI 파일을 **Tfrecord**의 형태로 변환하였습니다. 그 이유는 음원 데이터를 읽을 경우 데이터는 MIDI 파일로 저장되어 있고 이에 대한 메타 데이터와 레이블은 별도의 파일에 저장 되어 있기 때문에, 학습 데이터를 읽을 때 메타데이터나 레이블 파일 하나만 읽는 것이 아니라 MIDI 파일도 별도로 읽어야 하기 때문에, 코드가 복잡해지며 매번 읽어서 디코딩을 수행하게 되면 학습을 수행할 때 데이터를 읽는 부분에서 많은 성능 저하가 발생하게 됩니다. **TFRecord**를 사용하게 되면 이러한 문제점을 해결할 수 있습니다.

```
from magenta.scripts.convert_dir_to_note_sequences import convert_directory

# convert_directory 함수를 이용하여 midi 파일을 tfrecord 파일로 변환
root_dir = '/content/groove'
output_file = '/content/data/tfrecord'

convert_directory(root_dir, output_file, recursive=True)
```

3.2. 학습

```
Config.__new__.__defaults__ = (None,) * len(Config._fields)
CONFIG_MAP = {}
HParams = HParams

CONFIG_MAP['groovae_4bar'] = Config(
    model=MusicVAE(lstm_models.BidirectionalLstmEncoder(),
                   lstm_models.GrooveLstmDecoder()),
    hparams=merge_hparams(
        lstm_models.get_default_hparams(),
        HParams(
            batch_size=512,
            max_seq_len=16 * 4,  # 4 bars w/ 16 steps per bar
            z_size=256,
            enc_rnn_size=[512],
            dec_rnn_size=[256, 256],
            max_beta=0.2,
            free_bits=48,
            dropout_keep_prob=0.3,
        )),
```

4마디에 해당하는 드럼 샘플을 추출하는 것이 과제의 목표이기 때문에 config.py의 ‘groovae_4bar’을 사용하였고 하이퍼파라미터값은 그대로 유지하였습니다. 붉은색으로 표시된 부분에서 알 수 있듯이 인코더로는 논문에서 사용한 **BidirectionalLstmEncoder** 클래스를 사용하고 디코더로는 **GrooveLstmDecoder** 클래스를 사용합니다.

인코더의 경우는 512개의 은닉 노드를 가진 양방향 LSTM 레이어 1개가 사용되었고, 디코더의 경우에는256개의 은닉 노드를 가진 Groove LSTM 레이어 2개가 사용되었습니다.

논문에서는 Long-term sequence를 사용할 때 발생하는 **Posterior collapse** 문제를 해결하기 위하여 계층적 디코더를 사용하였지만 여기서는 4마디의 짧은 seqence를 사용하기 때문에 conductor 없이 학습을 수행합니다.

max_seq_len은 16*4 즉 4마디를 의미하고 z_size는 잠재벡터의 크기를 의미합니다.

4. 생성

```
from magenta.models.music_vae.trained_model import TrainedModel
import note_seq

model = TrainedModel(
    config=CONFIG_MAP['groovae_4bar'],
    batch_size=1,
    checkpoint_dir_or_path='/content/train')

# 4마디 샘플을 생성
generated_sequence = model.sample(n=1, length=16*4, temperature=0.5)
note_seq.sequence_proto_to_midi_file(generated_sequence[0], '/content/sample/4bar_sample.mid')
```

TrainedModel 클래스에서 checkpoint의 경로를 설정하고 학습된 모델로 부터 4마디의 드럼 샘플을 추출합니다.