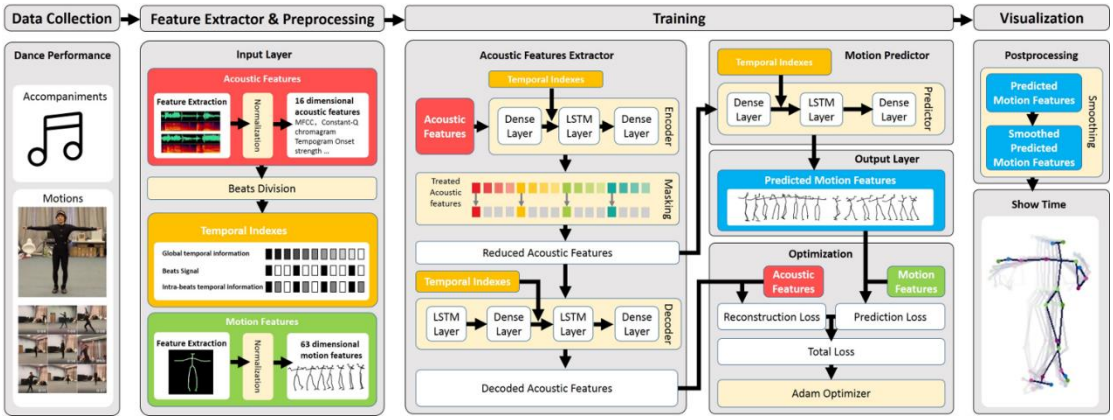
Report

1.1 Detail

1.1.1 Model



1.1.2 Dataset

https://github.com/Jarvisss/Music-to-Dance-Motion-Synthesis
数据集中共含有 4 种类型的舞蹈，Cha-cha，Tango，Rumba 和 Waltz。
共有 8 段 Cha-cha, 9 段 Tango，10 段 Rumba，34 段 Waltz。
其中音乐数据以 MP3 形式，动作数据以骨骼点的绝对位置形式保存。
其中 Tango、Rumba、Waltz 的动作数据帧率未知。Cha-cha 的帧率为 25fps。

1.1.3 Implementation

1. **Obtain data**
   a) 使用 Cha-cha 部分数据进行训练
   b) 数据集中包含 start/end_position，是由舞蹈人员给出开始/结束时间，通过 fps 计算。 其中 start、end 都是比较主观的，使用 librosa 重新提取节拍，然后让 start 等于原始 start 之后最近的一个拍，end 等于 start+动作的 frame length。
   c) 用新的 start，end 截取 music 帧，使之与 motion 帧对齐。
2. **Feature extraction**
   a) **Acoustic features**

Feature	Sound Characteristic	Definition
MFCC	Pitch	$a_i^1, \ldots, a_i^3$
MFCC-delta	Pitch change	$a_i^4, \ldots, a_i^6$
Constant-Q chromagram	Pitch	$a_i^7, \ldots, a_i^{10}$
Tempogram	Strength	$a_i^{11}, \ldots, a_i^{15}$
Onset strength	Strength	a_i^{16}

对齐后的 music 使用 librosa 提取 features。

i. Mfcc, mfcc_delta 人声

ii. Cqt_chroma 音调

iii. Onset_envelope 音量

iv. Tempogram 节拍周期

```
mel_spectrum =
librosa.feature.melspectrogram( y=self.music_data,sr=self.sr,n_fft=window_length,
hop_length=hop_length)
mfcc = librosa.feature.mfcc(S=mel_spectrum,n_mfcc=20) # mfcc[3]
mfcc_delta = librosa.feature.delta(mfcc) # mfcc_delta[3]
cqt_chroma = librosa.feature.chroma_cqt
(y=self.music_data,sr=self.sr,hop_length=hop_length,tuning=0,n_chroma=4)#cqt_chroma[4]
onset_envelope = librosa.onset.onset_strength(S=mel_spectrum) # onset_envelope[1]
tempogram = librosa.feature.tempogram(win_length=5, onset_envelope=onset_envelope)  #
tempogram[5]
```

b) **Temporal features**

Index	Definition
Arithmetic progression through the whole song	t_i^1
First frame of each beat	t_i^2
Arithmetic progression repeated within beats	t_i^3

使用 librosa.beat.beat_track()函数计算 beat，得到以上的 temporal feature

```
beats = librosa.beat.beat_track(y=self.music_data, sr = self.sr,hop_length=hop_length)
temporal_indexes_1 = np.array([i for i in range(mel_spectrum.shape[1])])
temporal_indexes_2 = np.array([1 if i in set(beats[1]) else 0    for i in range(mel_spectrum.shape[1])])
temporal_indexes_3 = np.array(temporal_indexes_2.copy())
in_frame_count=0
for i in range(len(temporal_indexes_3)):
    if temporal_indexes_3[i] == 1:
        temporal_indexes_3[i] = 0
        in_frame_count = 1
    else:
        temporal_indexes_3[i] = in_frame_count
        in_frame_count += 1
```

c) **Motion Features**

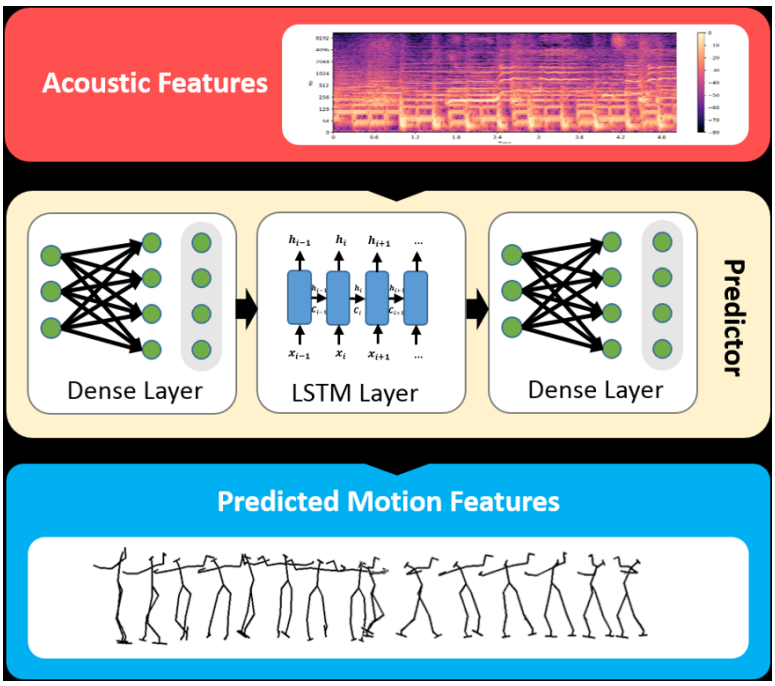
Motion feature 在数据集中已经给出，还需要做 center 和 rotate 的操作，在本做法中根据胯骨关节点计算模型朝向与垂直屏幕的面的夹角α，然后根据腰部中心点做地面的垂线 l，将整个模型绕 l 轴旋转α角度使其对齐。

d) **Normalize**

在 training set 上对 acoustic features 和 motion features ，对每个特征维度分别做 min-max normalize，测试时再用训练集上的 min-max 值进行还原。

3. Model implementation

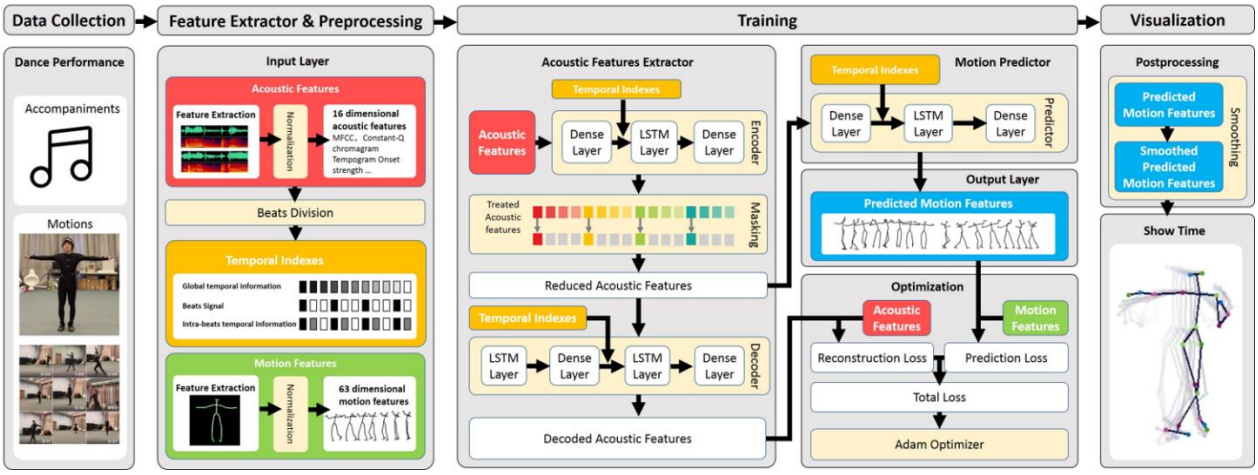
a) LSTM



虽然图上画的是 Acoustic features 作为输入，但是实际上论文中写的是 Acoustic features + temporal features 作为输入，经过全连接层增加模型的非线性，然后将编码后的 features 输入 3-layer LSTM，每个 time-step 输出 h_t，再通过全连接层预测动作序列 m_t。

文中没有对全连接层的深度、宽度、激活函数作任何描述，我在实现的过程中使用 2 层全连接，宽度是 64，Relu 的方式。

b) LSTM-autoencoder



根据 temporal index[0]，也就是 beat 信息，对 lstm-encoder 生成的序列进行

masking，得到 **Z**。

得到的 latent vector sequence **Z** 通过全连接层，与 temporal index 连接，输入到 decoder-lstm 中，再通过全连接层输入为 acoustic features，与原始 acoustic features 求 L2-loss，得到 loss1；

得到的 latent vector sequence **Z** 通过另一个 lstm 的预测网络，输出动作预测的序列，与 ground-truth 求 L2-loss， 得到 loss2.

Final_loss = max (threshold, loss1) + loss2. threshold 在实验中取 0.045.

1.1.4 Experiments

Epoch	Tempo	Rotate	Center	Leaky_relu	Maksing	Reduced size	Small_data	Result
9000						10		Good

1.1.5 Visualization Results

Results are available here

1.1.6 Problem

a) 不使用 temporal features，在小数据集上可以 train 到过拟合，但是用整段音乐输入则越到后面与 ground truth 的差别越大。

b) 使用 temporal features，不能 train 到过拟合，但是在预测序列中能够看出 beat 的感觉，长期预测结果则会趋于平均值。

c) 下一步需要进一步降低 loss 并提高模型的泛化能力，提高模型在预测长期序列上的准确度。

i. 增大数据集

ii. 使用 teacher-forcing 和 curriculum-forcing 的手段提高模型的

iii. 探究 VAE 代替 AE 的可能性。