# 2019.05.05

## 1.1 Summary

### 1. Music-to-dance 的再总结

(1) Preprocessing 部分

a) **提取音乐特征可以是任意维度的。**使用文中的 16 维(MFCC[3], MFCC_delta[3], cqt_chroma[4], tempogram[5], onset_strength[1]), 和直接取梅尔频谱 Mel_spectrum[80], 没有本质的区别, 因为上述 16 维特征均可以通过 Mel_sepctrum 提取。

b) **音乐、动作特征的归一化需要逐 Feature 的做，而不是逐维度做。**比如说上述的 16 维音乐特征 F, 不应该对每一维单独做归一化, 而应该对每个特征 (MFCC, MFCC_delta, cqt_chroma, tempogram, onsetstrength) 做归一化，也就是做 5 次归一化，而非 16 次。原因是对每一维做归一化相当于丢失了单个特征内的不同维度的差别信息，强行让特征内本身的差异无效。下图所示的 ablation study(除了 normalize 的方式不同以外，其他均一致)的 Loss 图佐证了这一点，可以看到 feature-scale normalize 的方式 Loss 下降的更快，更稳定，说明这种方式的 Normalize 更有效。Normalize 使用 min-max 好于 Z-score。

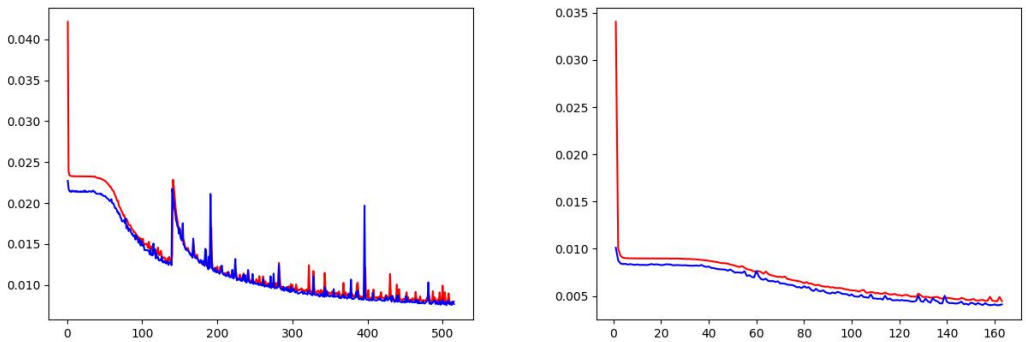



表 1 left: per **Dim** normalize, right: per **feature** normalize

Red: Train_loss, Blue:Valid_loss

c) **对 Feature 进行 Overlap 是必要的。**输入 Overlap 的数据可以有效的扩增数据集，不进行 overlap 甚至无法 cover 训练集的舞蹈。

d) **对 Music Clip 和 Dance Clip 进行对齐和剪辑是必要的。**对齐的意义在于使 LSTM 更容易学习到 Music-Motion 的 Mapping；剪辑的意义在于排除不连续的数据。

e) **对 Temporal Features 进行 normalize 是必要的。**意义在于防止 Temporal index 中的帧计数值过大，导致其他 feature 失效。

(2) Network 部分

a) Naïve LSTM without Masking, without temporal index

1) 模型难以收敛

2) 动作在一个节拍内不连续

b) Naïve LSTM without Masking, with temporal index

1) 模型收敛较慢

2) 节拍内舞蹈动作不连续

c) Naïve LSTM with Masking, with temporal index

1) 收敛较快

2) 节拍内舞蹈动作连续

3) Validation Loss 到后面很大

d) LSTM-AutoEncoder with Masking, with Masking, with temporal index

1) 收敛较快

2) 节拍内舞蹈动作连续

3) Validation Loss 和 Training Loss 同步下降

e) Other parameters / tricks:

1) *Adam optimizer*( learning rate=1e-3)

2) *LeakyReLU* instead of *ReLU* in MLP layers

3) *Orthogonal Init LSTM weights*

(3) PostProcessing 部分

a) Python OpenCV 画骨骼点

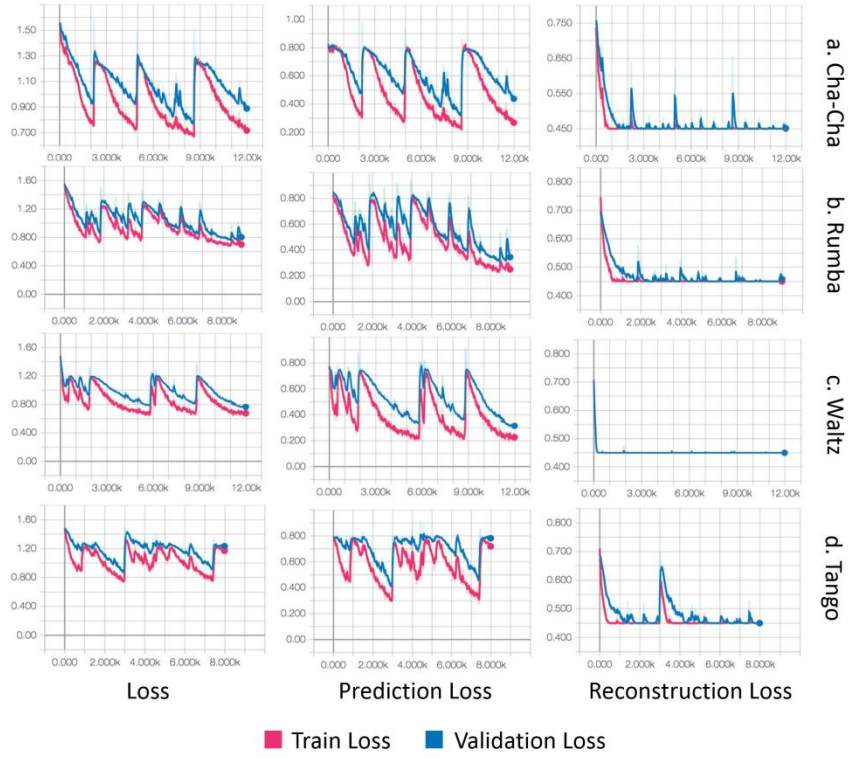
b) 使用 Moviepy 工具对 OpenCV 生成的图片加入音频帧，生成视频，需要保证音频和图片的帧数一致。

(4) 论文中的错误

a) 论文中提到的 metric_loss 没有使用过。

$$metric\_loss = \sum_{t \in frames} \sqrt{\sum_{i \in frame[t]} \left(X_{t,i} - X'_{t,i}\right)^2} \qquad (10)$$

b) 图中 C 行第三列 Reconstruction Loss 没有画出 Train Loss




c) 对比实验图（下图）中 Train Loss 比 Validation Loss 大，而上图曲线中则 Train Loss 比 Validation Loss 小。