节奏类虚拟现实游戏的关卡生成

学士论文

Mariia Rizhko

由 Andrii Boichuk 指导

利沃夫，2022年春季

摘要

Ragnarock 是一款虚拟现实（VR）节奏游戏，玩家将扮演一名维京船长，参加长船竞赛。玩家需要用两把锤子，随着史诗般的维京音乐节拍敲碎迎面而来的符文。符文由节拍图谱定义，玩家可以手动创建节拍图谱。创建节拍图谱需要数小时。

本研究旨在自动化节拍图创建流程，也就是编排学习任务。该任务分为三个部分：确定节拍的时间点（动作放置）、确定与所选节拍相关的符文应放置在空间中的位置（动作选择），以及 Web 应用程序的创建。

对于动作放置这一首要任务，我们从音乐录音中提取主导局部脉冲（PLP）信息。此方法有助于学习节拍应放置的位置和数量。

对于动作选择这一第二项任务，我们使用循环神经网络（RNN）[10][11]，具体来说是门控循环单元（GRU）[12][13]，以学习节拍序列及其模式，从而能够重现这些规则并生成全新关卡。

最后一项任务是为非技术背景的玩家构建解决方案，将第一部分和第二部分的成果整合到一个便于使用的 Web 应用程序中。为完成这项任务，前端使用 JavaScript 和 React 构建，后端则使用 Python 和 FastAPI。

目录

1 引言 5

2 相关研究 9

2.1Dance Dance Revolution 9 2.2节奏光剑 9

3 材料与方法 11

3.1 构建数据集 11 3.2 数据处理 14

4 实验 16

4.1 动作放置 16 4.2 动作选择 23 4.3 应用程序 27

5 个结果 31

缩略语列表

VR - 虚拟现实 RNN - 循环神经网络
CNN - 卷积神经网络 GRU - 门控循环单元 MFCC - 梅尔频率倒谱系数 DFF - 离散傅里叶变换 DCT - 离散余弦变换
MLSTM- 乘法长短期记忆网络 PLP - 主导局部脉冲 JSON - JavaScript 对象表示法

1 引言

Ragnarock [1] 是一款节奏类VR游戏，于2021年年底发行，如今越来越受欢迎。游戏内置约20个关卡，还可以额外购买七首歌曲。玩家很快就能熟练掌握这些关卡，并希望玩到新歌曲，这一点不足为奇。虽然游戏允许玩家使用自定义地图，但手动创建地图可能需要数小时甚至一整天。

游戏的目标是随着音乐节奏挥动两把锤子，在符文到达鼓面时将其击碎。图1展示了Ragnarock的游戏画面。



图1：Ragnarock的游戏画面

这款游戏并非同类游戏中的首创。在它之前，已经有一款《节奏光剑》[2]。它是 2019 [7]最受欢迎的VR游戏。游戏描绘了用户在超现实的霓虹氛围中，挥舞一对颜色对比鲜明的光剑，切开代表音乐节拍的方块。

玩家使用虚拟现实动作控制器操控一对剑，左手为红色，右手为蓝色。每首歌曲都会有一连串迎面而来的方块。每个方块都会按照音乐节拍，在特定时刻出现。它可以出现在4×3网格中的12个位置之一。每个方块上都标有一个指向八个不同方向之一的箭头。玩家也可以从任意方向击打带圆点的方块。当光剑按照要求切中方块时，方块便会被摧毁，并根据切割长度给予分数。

这款游戏的目标与 Ragnarock 类似，但你不是击碎符文，而是切割立方体。图 2 展示了节奏光剑的游戏画面。

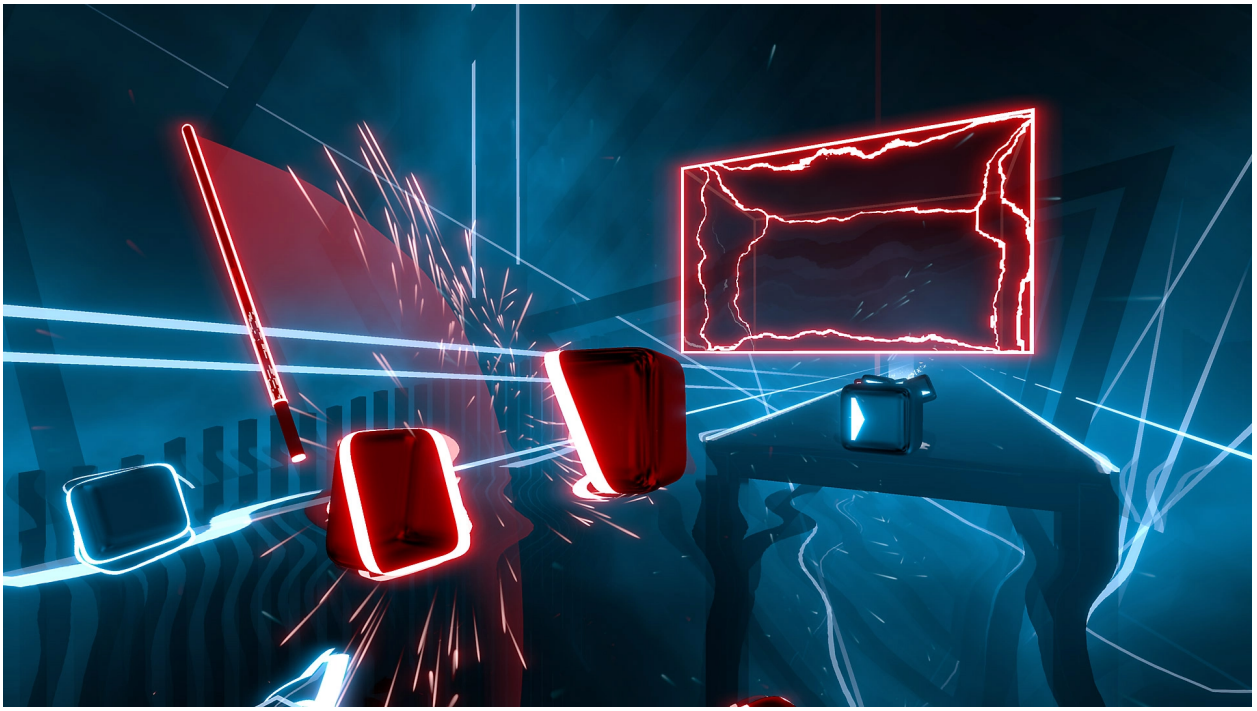

图 2：节奏光剑的游戏画面

另一款玩法相似但历史悠久得多的游戏是 Dance Dance Revolution。它是电子游戏中开创节奏与舞蹈类型的系列作品，1998年首次在日本发行，随后在北美和欧洲发行

于1999年。玩家根据音乐和视觉提示，站在“跳舞平台”或舞台上，用脚踩下排列成十字形的彩色箭头。系统会根据玩家配合给定节奏完成舞步的准确度进行评分；若玩家通过关卡，还会获得更多音乐曲目供其游玩。因此，在这里，玩家需要踩箭头，而不是击碎符文或切开立方体。图3展示了 Dance Dance Revolution 游戏玩法的截图。

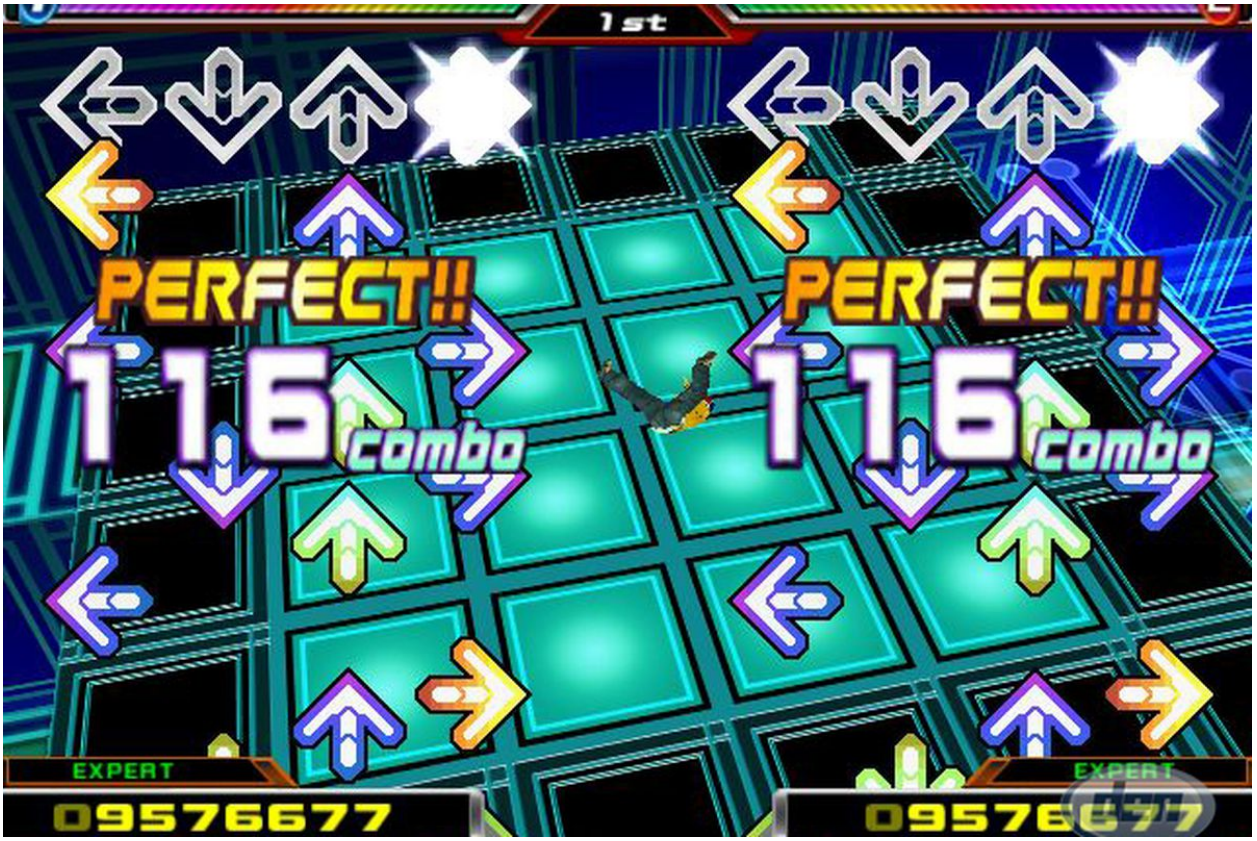

图 3：Dance Dance Revolution 的游戏画面

尽管这些游戏的机制和元素各不相同：在 Dance Dance Revolution 中，玩家需要踩踏箭头；在节奏光剑中，玩家需要斩击立方体；在 Ragnarock 中，玩家需要击碎符文，但关卡的核心理念相同：特定元素会在特定时间出现。这项任务最初是在 Dance Dance Convolution [3] 中作为编排学习任务提出的。

这项工作的目标是为 Ragnarock 学习编舞。该任务分为三个子任务。第一个是动作放置，需要生成元素放置时间的时间戳序列。第二个是动作选择，需要确定具体要放置哪些元素。最后一个是 Web 应用开发。

本工作的主要贡献是解决 Ragnarock 游戏的编排学习问题，并为非技术用户开发一款 Web 应用程序。

2 相关研究

2.1 Dance Dance Revolution

Dance Dance Convolution [3] 是首个尝试解决节奏类游戏关卡生成问题的工作。他们将其定义为编排学习任务，设计并评估了多种深度学习方法，还引入了标准化数据集和可复现的评估方法。

他们将任务分解为两个子任务：确定何时放置步点，以及选择哪些步点。对于步点放置任务，他们结合循环神经网络和卷积神经网络，输入低级音频特征的频谱图，以预测受谱面难度影响的动作。他们提出了一种用于步点选择的条件长短期记忆网络[15] 生成模型，其性能显著优于n元语法和固定窗口方法。

他们的处理流程如下：

1. 将音频表示为特征 2. 将步骤 1 得到的表示输入步点放置算法，估计步点位于该帧内的概率 3. 将步骤 2 得到的概率序列输入峰值检测算法，确定放置步点的精确时间戳 4. 使用步点选择算法，根据给定的时间戳序列，选择在每个时间点放置哪些步点

2.2 节奏光剑

DeepSaber：面向高维编舞的深度学习 [4]基于 Dance Dance Convolution 的研究成果。介绍了多种节奏光剑谱面生成

方法，并对其进行了描述、实现和评估。研究展示了自然语言与节拍图谱之间的相似性，并利用这一点推进节拍图谱生成技术。他们创建了一个动作类比数据集，并用它展示动作嵌入的高质量。他们利用动作嵌入开发了一种新的训练指标和一种基于新颖性的指标，用于比较完全生成的节拍图谱与人工创建的地图。

在音频处理中，他们使用梅尔频率倒谱系数（MFCC）[14] 特征，进行预加重，执行离散傅里叶变换（DFF），应用 40 个三角形梅尔滤波器组，取对数，计算离散余弦变换（DCT），取前 13 个倒谱系数，将其替换为帧能量的对数以考虑音量，最后移动时间戳。

《节奏光剑》中输入的维度数量差异很大。每种输入在表达抽象表示时的最佳参数数量也应有所不同。因此，他们采用了多重LSTM（MLSTM）架构，以找到能够应对输入维度差异的良好架构。由于LSTM块之间的连接是随机的，大多数架构产生的结果都不理想。他们通过在Hyperband算法中使用更多训练轮次（110轮）解决了这个问题。

3 材料与方法

3.1 构建数据集

游戏Ragnarock允许玩家手动创建地图，因此社区创作了数百张地图，并将它们发布在网站 [5]上。该网站的截图见图4。

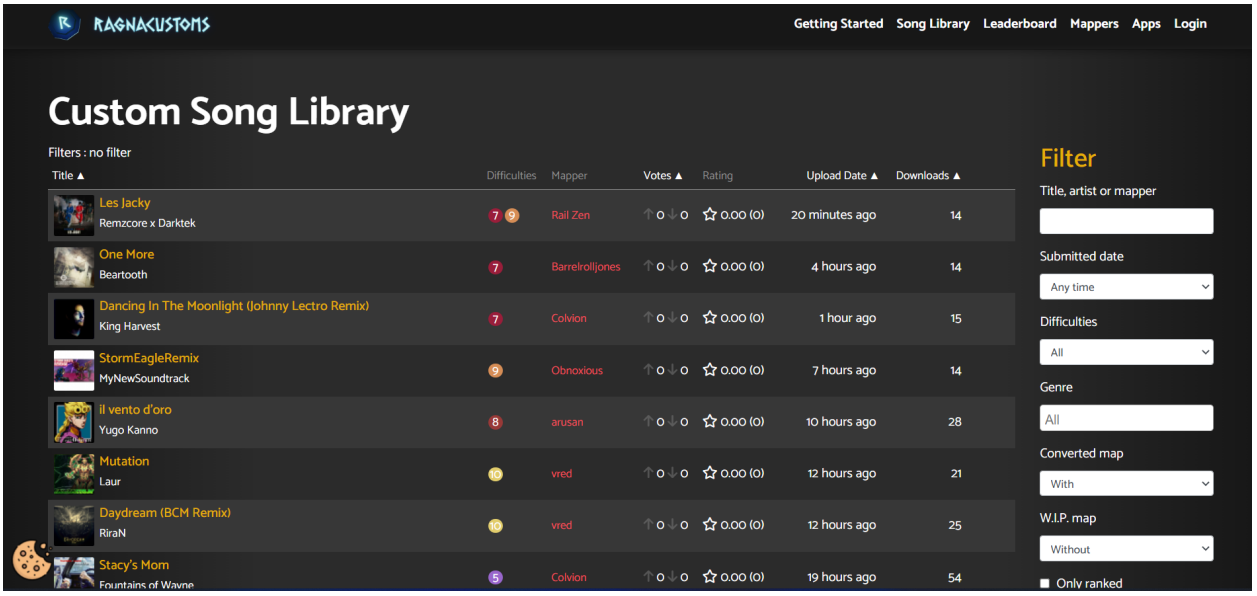

图 4：手动创建关卡的网站

为了创建数据集，我们解析了该网站上的关卡。总计有 831 首歌曲。对于每首歌曲，解析了以下数据：歌曲编号、歌曲名称、歌曲作者、歌曲文件名、每分钟节拍数（BPM），以及是否存在简单、普通、困难、专家或专家+难度。

不同难度的地图数量各不相同，具体如下：

- 简单：572 - 普通：130 - 困难：95 - 专家：147 - 专家+：186。

每个游戏关卡都以文件夹的形式呈现，用户可以将其放入游戏中并播放歌曲。该文件夹中包含以下文件：

- Info.dat：包含关卡一般信息的必需文件 - 歌曲文件：必需的 .ogg 格式音频文件 - 歌曲预览文件：歌曲文件的必需短版本，同样为 .ogg 格式 - 封面图像文件：关卡预览所需的必备图片 - Easy.dat：简单难度的可选文件 - Normal.dat：普通难度的可选文件 - Hard.dat：困难难度的可选文件 - Expert.dat：专家难度的可选文件 - ExpertPlus.dat：专家+难度的可选文件

Info.dat 文件包含关卡的一般信息，例如：

- 歌曲名称 - 歌曲副标题 - 歌曲作者姓名 - 关卡作者姓名 - 每分钟节拍数 - 歌曲文件名 - 封面图像文件名 - 难度节拍图集合 - 等等

difficultyBeatmapSets 是一个难度节拍图集合列表。每个集合都包含以下信息：

- 难度

- 难度等级 - 谱面文件名 -
音符跳跃移动速度 - 音符跳
跃起始节拍偏移

beatmapFilename 是关卡难度谱面文件的名称（例如 Easy.dat、Normal.dat、Hard.dat 等）。该难度谱面是一个字典，其中一个键是 _notes。_notes 是一个节拍列表，每个节拍包含：

- 时间_- 轨道索
引_- _轨道层 -
_类型 - 切割方
向_

图5展示了此类文件的示例。


```
▼ _notes: [] 679 items
 ▼ 0:
    _time: 4
    _lineIndex: 0
    _lineLayer: 1
    _type: 0
    _cutDirection: 1
 ▼ 1:
    _time: 4
    _lineIndex: 1
    _lineLayer: 1
    _type: 0
    _cutDirection: 1
 ▼ 2:
    _time: 4
    _lineIndex: 2
    _lineLayer: 1
    _type: 0
    _cutDirection: 1
 ▼ 3:
    _time: 4.5
    _lineIndex: 1
    _lineLayer: 1
    _type: 0
    _cutDirection: 1
```


图5：beatmapFilename 示例

动作放置的第一项任务是为每个节拍设置 _time 参数。动作选择的第二项任务是设置_lineIndex,

_lineLayer、_type 和 _cutDirection。除此之外，应用创建的最后一项任务是根据用户选择设置所有其他参数，将文件保存在一个文件夹中，并发送给用户。

3.2 数据处理

如第 3.1 节所述，文件以字典的形式呈现，这意味着在将其用于任何机器学习算法之前，需要进行额外的预处理。

每个谱面文件名都会被解析为一个元素列表。每个元素根据节拍的 _lineIndex、_lineLayer、_type 和 _cutDirection 表示一个唯一标识符。该唯一标识符以三乘四网格的扁平表示形式构建。具体分为以下几个步骤:

1. 选择至少包含一个元素的时间戳 2. 为该元素赋默认值 [['n9', 'n9', 'n9', 'n9'], ['n9', 'n9', 'n9', 'n9'],['n9', 'n9', 'n9', 'n9']] 3. 将网格中位置（_lineLayer, _linIndex）处的值改为 _type + _cutDirection，其中 + 是字符串拼接操作 4. 如果同一时间戳还有其他元素，则对这些元素重复步骤 3 5. 将数组展平为字符串

预处理的结果如图6所示。

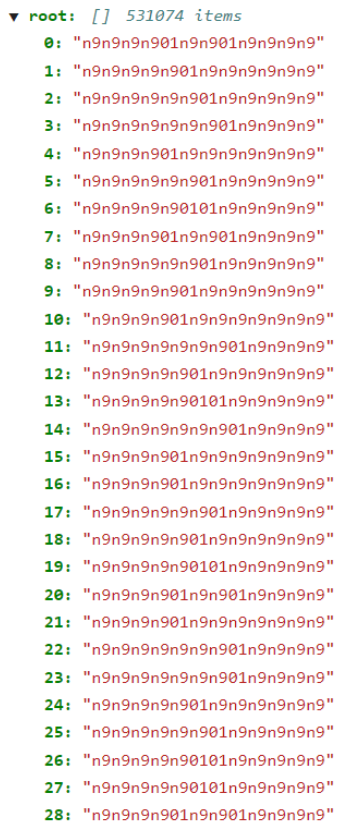

图6：预处理后的谱面文件名示例

4 实验

4.1 动作放置

任务是根据原始音频，找出放置节拍的时间戳，如图 7 所示。

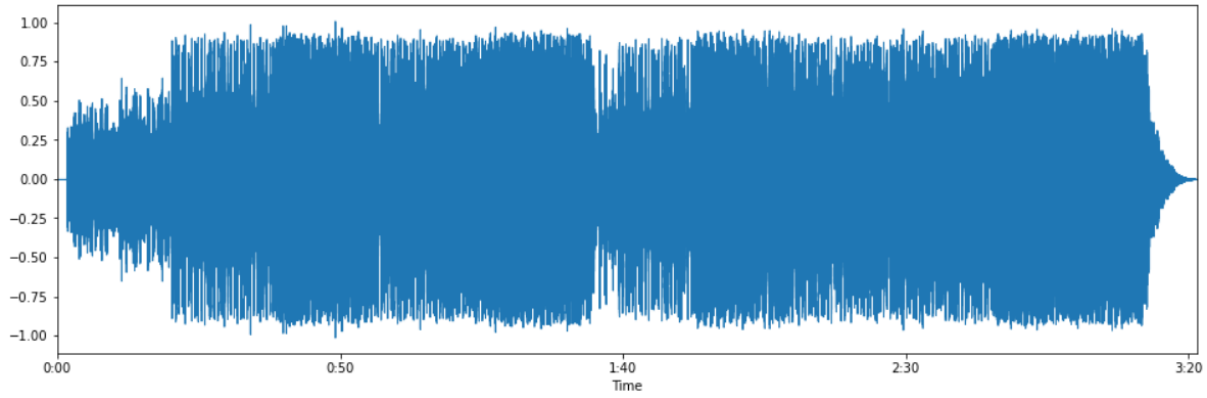

图 7：原始音频文件

由于 Ragnarock 是一款节奏类游戏，大多数节拍的位置取决于音乐曲目的节奏。因此，从理论上讲，歌曲的动态脉冲与节拍位置之间应存在相关性。这里使用主导局部脉冲（PLP）[6][8][9] 估计。PLP 方法会在频域中分析起音强度包络，为每一帧找出局部稳定的速度。这些局部周期性会合成局部半波，并以峰值与节奏显著帧（例如，音乐时间网格上的起音事件）重合的方式组合起来。脉冲曲线的局部极大值可作为估计的节拍位置。

图 8 展示了将 PLP 应用于一个包含 1000 帧的音频样本，再对 PLP 应用局部极大值算法的示例。

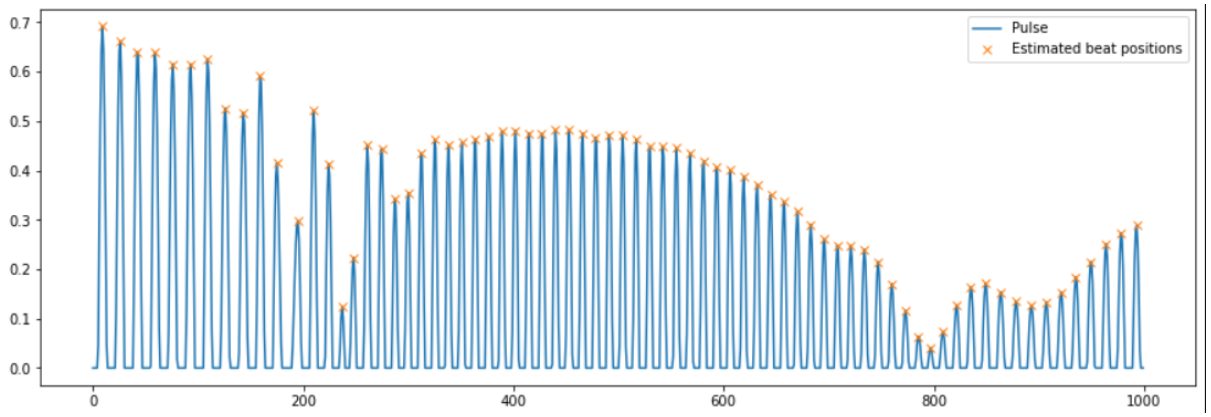

图 8：PLP 和估计的节拍位置

蓝线表示局部脉冲随帧的变化。橙色叉号表示通过对 PLP 应用局部极大值算法找到的估计节拍位置。

估计的节拍位置是理想解，但实际数据有所不同。图 9 显示了 PLP 上的实际节拍位置。

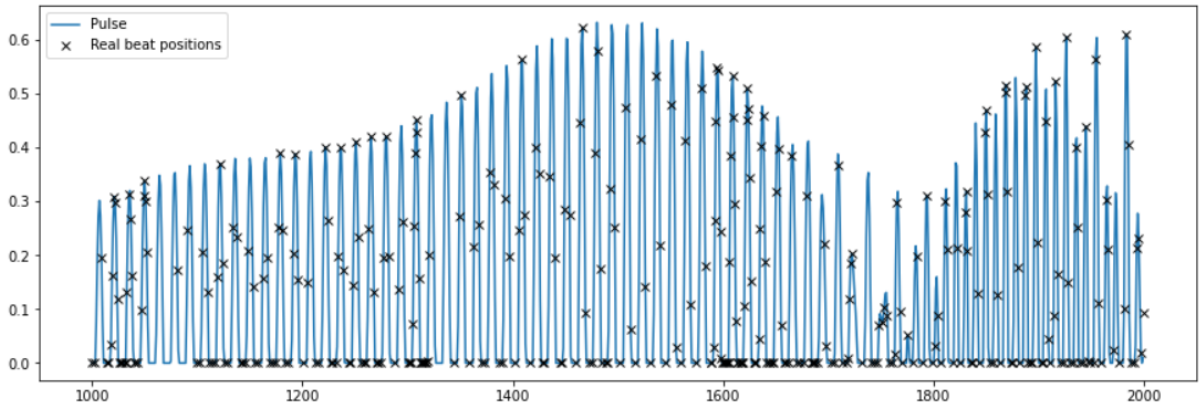

图 9：PLP 和实际节拍位置

随着 PLP 变化，数据总体呈下降趋势，但并不理想。采用次数为 3 的一维平滑样条拟合估算数据和实际数据，以验证估算节拍位置与实际节拍位置之间的相关性。样条显示在图 10 中。

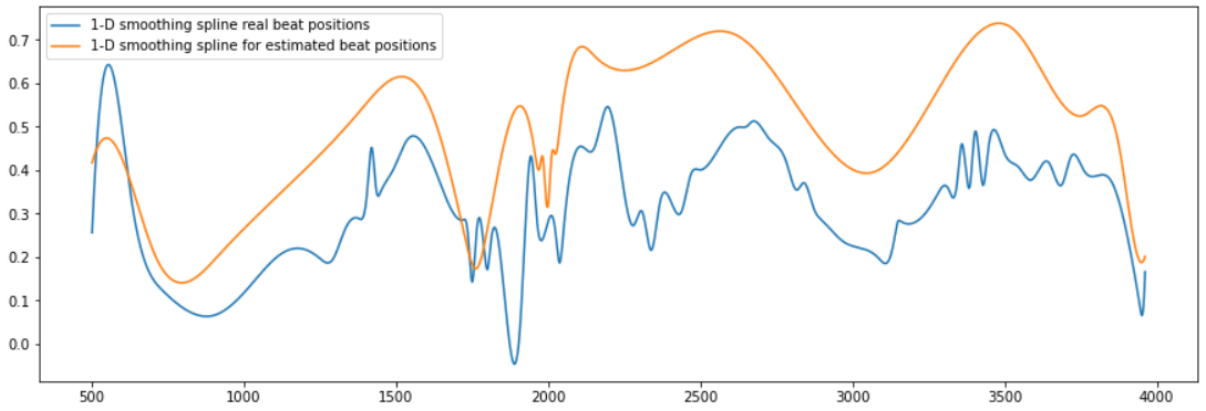

图 10：样条拟合实际节拍位置和估计的节拍位置。

从上图可以推断，估算节拍位置与实际节拍位置之间存在直接的正相关。这两条样条的相关系数为 0.75，这意味着 PLP 和随后的局部极大值搜索是适用于动作放置任务的良好算法。

另一个需要回答的问题是应放置多少个节拍。游戏关卡越高，出现的节拍就应该越多。通过调整距离参数，可以改变局部极大值搜索中选出的峰值数量。相邻峰值之间要求的最小水平距离必须大于或等于 1 个样本。系统会先移除较小的峰值，直到所有剩余峰值都满足这一条件。距离越大，选出的峰值数量越少。图 11 展示了减少节拍数量的示例。

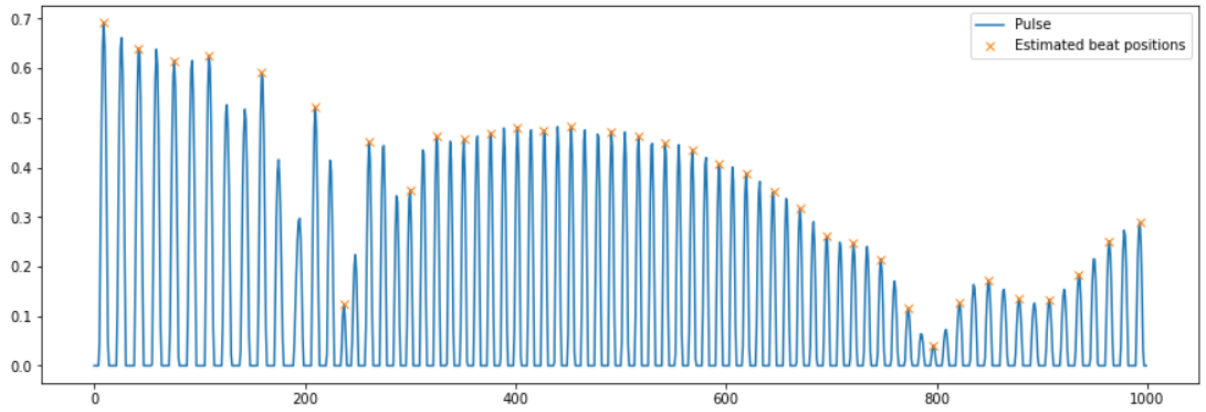

图 11：PLP 与减少后的估计节拍位置数量

为了确定不同关卡的最优距离，我们为每种难度选取了十个样本，并记录了对应的歌曲时长、信息中提供的每分钟节拍数、librosa 库计算的每分钟节拍数以及节拍数量。所选数据如图 12–16 所示。

	diff	duration	bpm_info	bpm_librosa	num_el
0	Easy	190.827256	102.0	103.359375	505
1	Easy	124.226757	120.0	112.347147	362
2	Easy	258.647075	320.0	117.453835	667
3	Easy	198.089478	130.0	129.199219	656
4	Easy	237.621451	82.0	161.499023	823
5	Easy	189.141088	105.0	107.666016	628
6	Easy	516.585986	96.0	129.199219	2053
7	Easy	273.501270	98.0	99.384014	608
8	Easy	269.545034	87.0	172.265625	764
9	Easy	243.577370	140.0	143.554688	897

	diff	duration	bpm_info	bpm_librosa	num_el
10	Normal	233.887392	91.0	92.285156	781
11	Normal	254.189841	135.0	135.999178	1463
12	Normal	230.365170	88.0	89.102909	671
13	Normal	269.373333	178.0	135.999178	1929
14	Normal	271.037732	213.0	107.666016	1142
15	Normal	190.721451	174.0	117.453835	627
16	Normal	118.767528	148.0	143.554688	284
17	Normal	136.080590	126.0	129.199219	566
18	Normal	150.793016	130.0	129.199219	717
19	Normal	262.452245	160.0	161.499023	458

图 12–13：简单和普通难度级别的选定数据

	diff	duration	bpm_info	bpm_librosa	num_el
20	Hard	132.021905	150.000000	151.999081	412
21	Hard	168.741950	155.000000	103.359375	655
22	Hard	234.979955	105.000000	103.359375	1014
23	Hard	164.633651	174.000000	172.265625	536
24	Hard	214.025079	86.000000	117.453835	897
25	Hard	229.465079	100.000000	99.384014	703
26	Hard	230.135011	220.000000	112.347147	859
27	Hard	191.232336	106.248001	107.666016	517
28	Hard	549.760000	108.000000	107.666016	2093
29	Hard	218.600952	130.000000	129.199219	833

	diff	duration	bpm_info	bpm_librosa	num_el
30	Expert	156.639819	164.000000	107.666016	621
31	Expert	192.000000	200.000000	99.384014	594
32	Expert	164.193741	152.000000	151.999081	459
33	Expert	121.203084	91.500000	92.285156	322
34	Expert	217.646168	130.106003	129.199219	600
35	Expert	155.421315	105.000000	143.554688	703
36	Expert	201.314286	178.000000	117.453835	961
37	Expert	278.502948	125.000000	129.199219	755
38	Expert	317.504989	125.000000	123.046875	1258
39	Expert	264.639048	124.000000	123.046875	1392

图14–15：困难和专家难度级别的选定数据

	diff	duration	bpm_info	bpm_librosa	num_el
40	ExpertPlus	339.888254	130.0	129.199219	1721
41	ExpertPlus	298.460544	127.0	129.199219	1229
42	ExpertPlus	249.460862	159.0	161.499023	916
43	ExpertPlus	282.031338	200.0	99.384014	1863
44	ExpertPlus	215.088390	135.0	135.999178	945
45	ExpertPlus	195.000045	138.0	135.999178	952
46	ExpertPlus	238.205079	94.0	123.046875	1287
47	ExpertPlus	197.435828	130.0	129.199219	1135
48	ExpertPlus	299.742041	260.0	129.199219	2237
49	ExpertPlus	186.328118	190.0	95.703125	812

图16：专家+难度级别的选定数据

为找出 num_el 与其他参数之间的依赖关系，绘制了以下图表：图17中的 num_el / 时长、图18中的 num_el / 每分钟节拍数_librosa、图19中的 num_el / 每分钟节拍数_info，以及图20中的 num_el / (时长 *每分钟节拍数_info)。

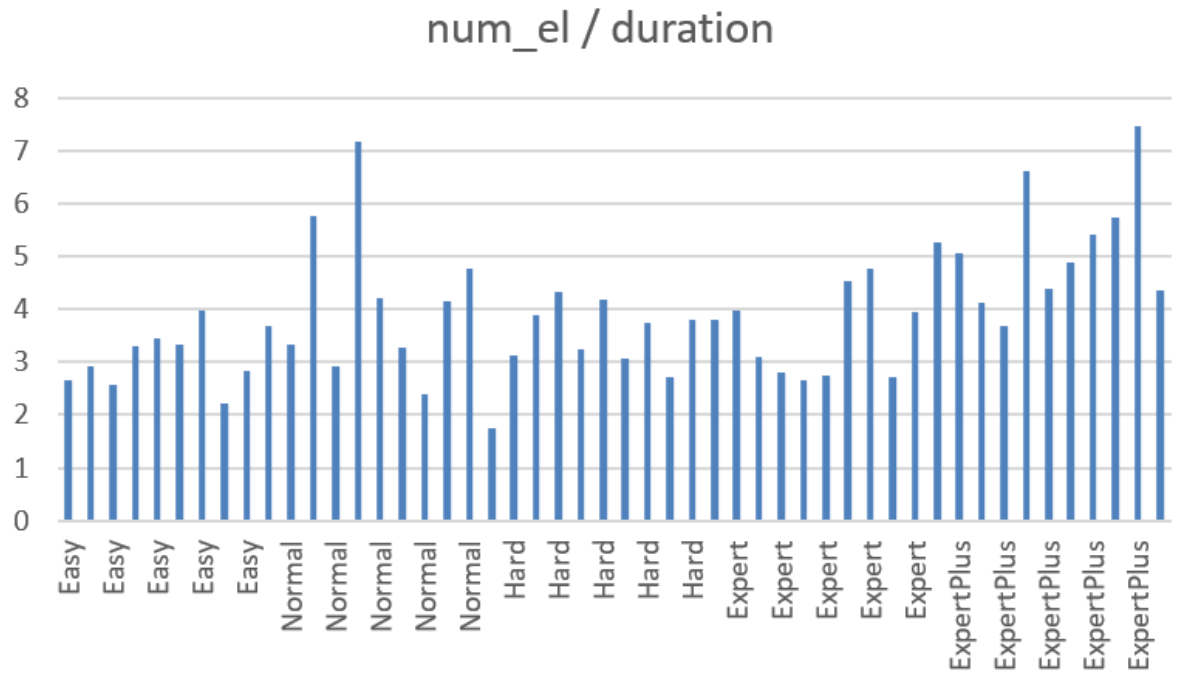

图17：Num_el 与时长的相关性

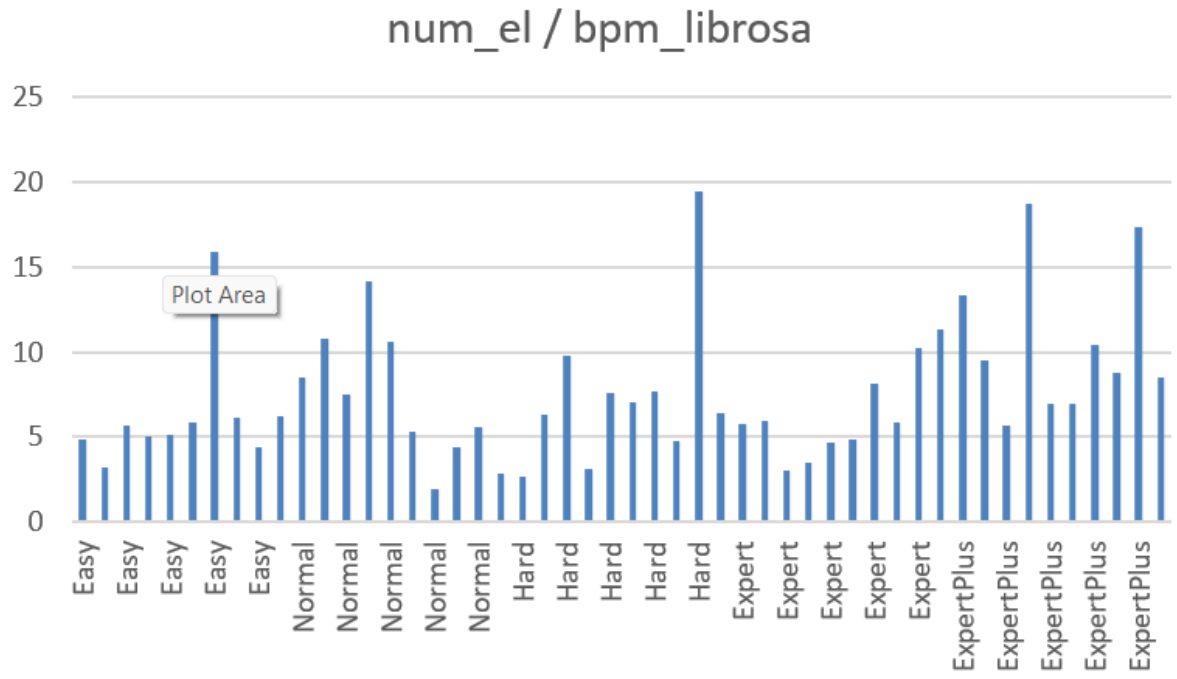

图18：Num_el 与 bpm_librosa 的相关性

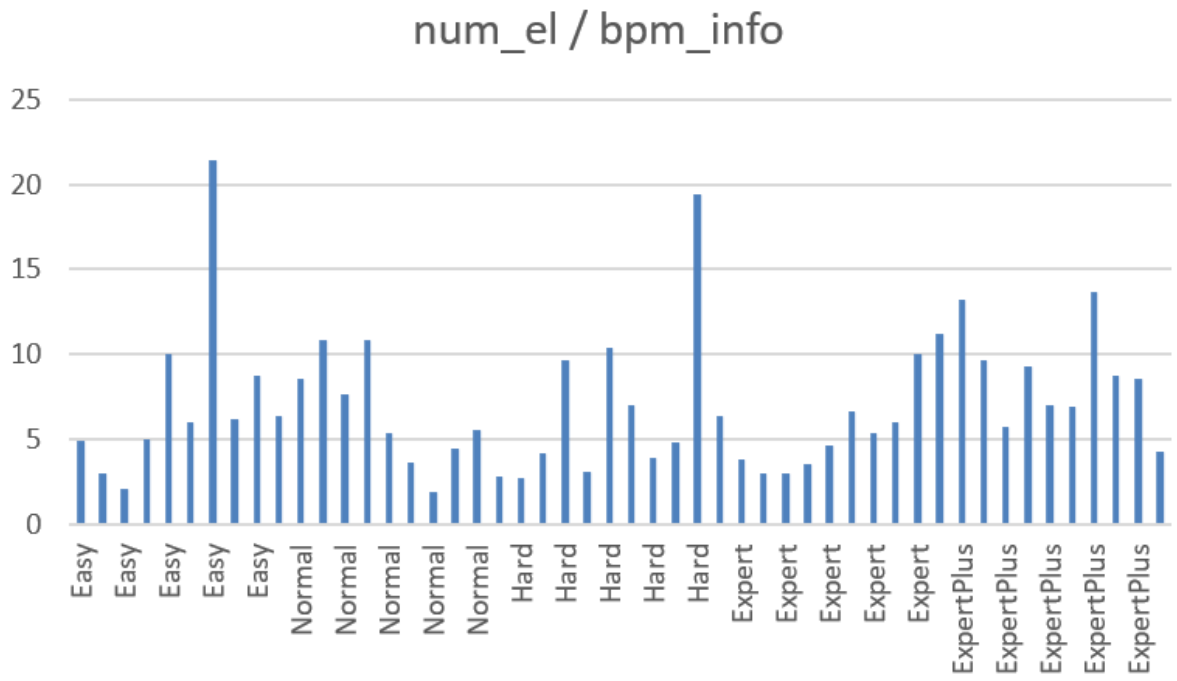

图19：Num_el 与 bpm_info 的相关性

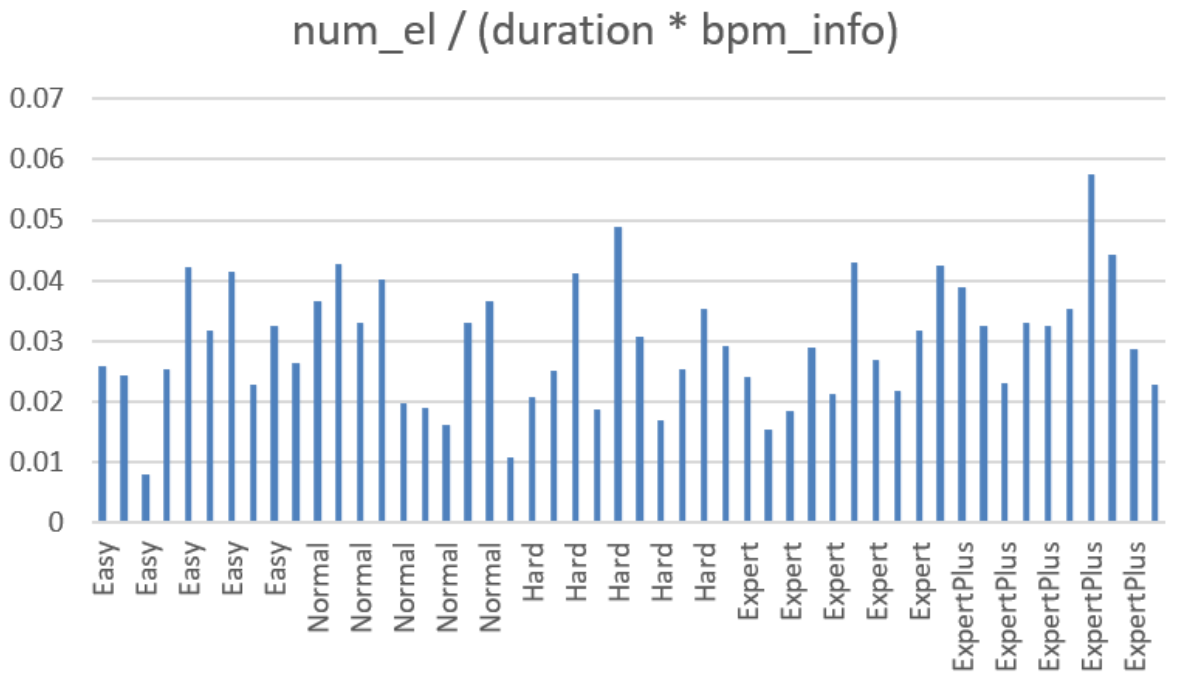

图20：Num_el 与 duration * bpm_info 的相关性

从上面的图表中很难直观看出不同关卡之间的数值差异，因此针对每种难度，计算了 num el / 时长的最小值、最大值和平均值。结果如表 1 所示。_

难度	Min	Max	平均值
Easy	2.223024	3.974169	3.093952
普通	1.745079	7.161065	3.851733
Hard	2.703518	4.315262	3.60789
专家	2.656698	5.259995	3.572549
专家+	3.671919	7.463084	5.071498

表 1：num_el / 时长的最小值、最大值和平均值

因此，简单关卡每秒应有约 3.09 个节拍，普通、困难和专家关卡约为 3.65 个，而专家+关卡约为 5.07 个。实际应用中，选择最大峰值数量（距离 = 1）可以为任何难度的关卡生成合适数量的节拍。

4.2 动作选择

对于动作选择任务，由于组合不会随着难度级别的提高而变得更加复杂，因此将所有难度级别合并。预处理数据的总长度为 915012 个元素。总共有 15 个唯一元素，并将它们保存到一个数组中。也就是说，在任意时间戳，都可能同时出现这 15 个元素之一，它们代表 0、1、2 个或更多符文。每种可能元素的出现频率见表 2。

元素	数量
n9n9n9n901n9n901n9n9n9n9	37389
n9n9n9n9n9n9n901n9n9n9n9	147836
n9n9n9n9n9010101n9n9n9n9	4
n9n9n9n901n90101n9n9n9n9	2
n9n9n9n9n9n90101n9n9n9n9	23993
n9n9n9n9n90101n9n9n9n9	48583
n9n9n9n9n901n901n9n9n9n9	27323
n9n9n9n9n9n9n9n9n9n9n9n9	28159
n9n9n9n9n9n901n9n9n9n9n9	197870
n9n9n9n90101n9n9n9n9n9n9	26163
n9n9n9n9n901n9n9n9n9n9n9	200806
n9n9n9n901n901n9n9n9n9n9n	26007
n9n9n9n901010101n9n9n9n9	2
n9n9n9n901n9n9n9n9n9n9n9	150871
n9n9n9n9010101n9n9n9n9n9	4

表 2：预处理数据中的元素频率

表中有 4 个元素在数据中出现的次数不超过 4 次。这些属于异常值，因为游戏内同一时间戳最多只能出现两个符文。玩家只有两把锤子，因此不可能同时击碎两个以上的符文。因此，这些元素已从数据中移除。

元素 ‘n9n9n9n9n9n9n9n9n9n9n9’ 表示该时间戳处没有符文。由于可以从_笔记中移除此元素，且结果不变，因此它是多余的。所以也移除了这个元素。

该任务被视为文本生成任务，但使用的是 10 种唯一元素，而不是 26 个字母。为了将这段庞大的数据串（915012 个元素）转换为输入，数据被拆分为大小为 200 的数据块。

该模型以当前步骤的字符作为输入，并预期t_{-1}输出下一个字符。模型包含三层：一个将t输入字符编码为内部状态的线性层；一个作用于该内部状态的 GRU 层，其隐藏状态大小为 100；以及一个用于输出概率分布的解码层。

每个数据块通过遍历其中的元素，并查找每个元素在中的索引，转换为张量。最后，从一个随机数据块组装出一$all_characters$对用于训练的输入张量和目标张量。输入包含除最后一个元素外的所有元素，目标则包含除第一个元素外的所有元素。因此，如果数据块是 "123"，输入对应 "12"，而目标对应 "23"。

评估网络时，每次输入一个字符，并将网络的输出用作下一个字符的概率分布，以此类推。开始生成时，会传入一个提示字符串，以构建隐藏状态；随后，网络每次生成一个字符。

然后定义其他训练参数：

- 训练轮次：2000 - 初始学习率：
0.0001 - 解码器优化器：
Adamoptimizer

- 损失函数：交叉熵损失 - 调度器：指数学习率调度器，伽马参数为 0.995

训练过程中损失变化的曲线见图 21，学习率变化的曲线见图 22。

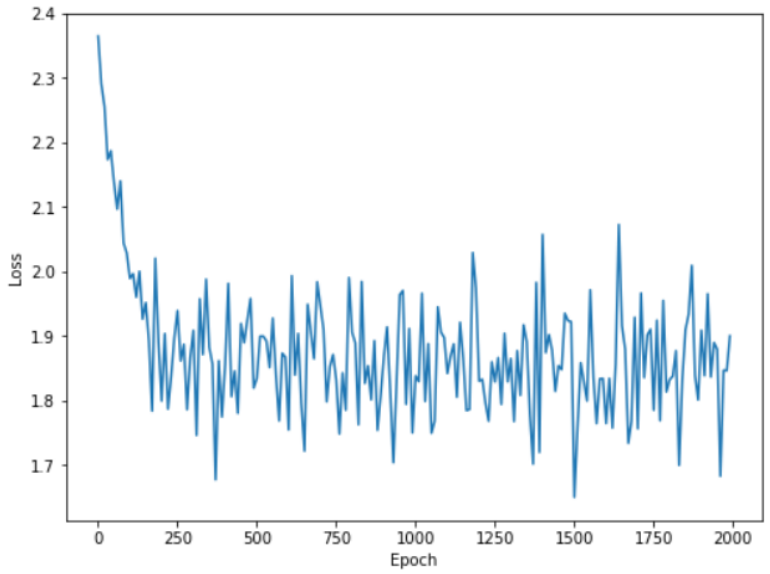

图 21：训练期间的损失变化

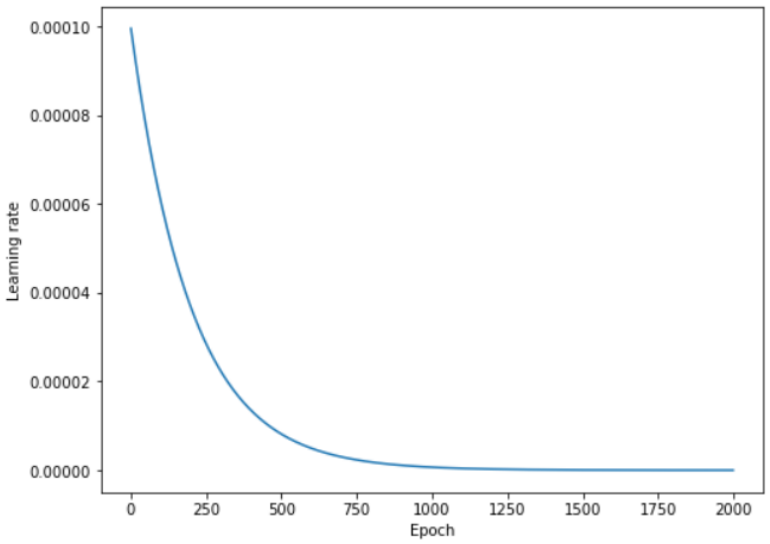

图 22：训练期间的学习率变化

如图 22 所示，学习率在每次迭代时都会降低。较大的初始学习率有助于模型更快地学习，而较小的末期学习率则有助于模型找到更优甚至全局最优的解。如图 21 所示，经过 250 个训练轮次后，损失没有显著变化，这意味着模型已经学到了主要模式。

训练完成后，将使用 evaluate 函数。每次进行预测时，都会将输出除以传入的“温度”参数。数值越大，所有动作发生的可能性就越接近，因此输出会“更加随机”。数值越小（小于 1），高概率的影响就越大。当温度趋近于零时，只会选择概率最高的输出。

为了在 Web 应用程序中进一步使用，训练好的模型被保存到“model.pth”文件中，唯一元素被保存到“all_characters.json”中，参数（hidden_size 和 n_layers）被保存到“params.json”中。

4.3 应用程序

模型本身不便于普通用户使用，因为使用它需要一定的技术技能。为了解决这个问题，我们创建了 Web 应用程序。它由两部分组成：前端和后端。其工作处理流程如下：

1. 用户访问一个网站。图 23 展示了该网站的屏幕截图。2. 用户点击“上传文件”按钮并选择要使用的歌曲。3. 用户选择不同的难度等级。正如 Web 应用程序中所述，可选的难度等级如下：- 简单：等级 1-2 - 普通：等级 3-4

- 困难：等级 5-6 - 专家：等级 7-8 - 专家+：等级 9-10

4. 用户点击“下载文件”按钮 5. 用户解压下载的文件夹，并将其放入游戏中

步骤 2、3 和 4 如图 24 所示。

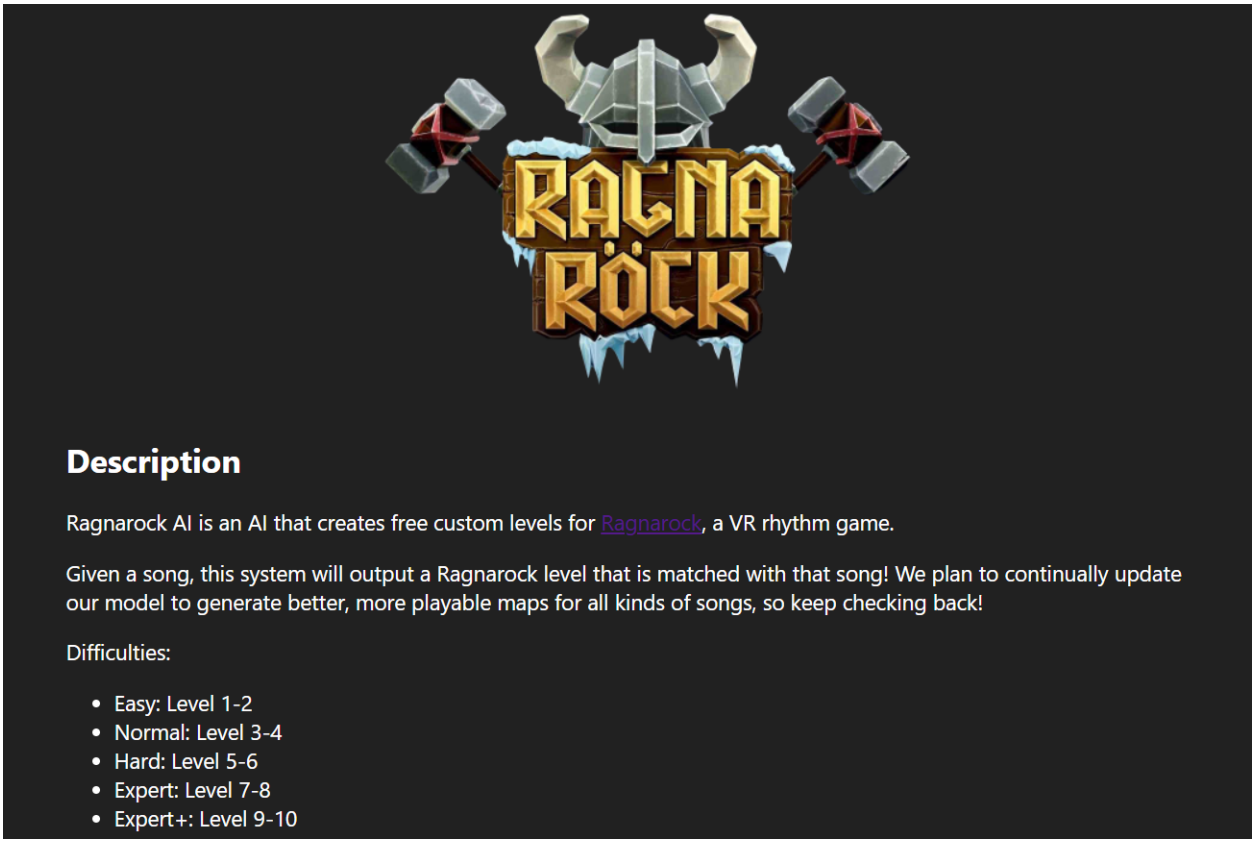

图 23：Web 应用程序第 1 部分

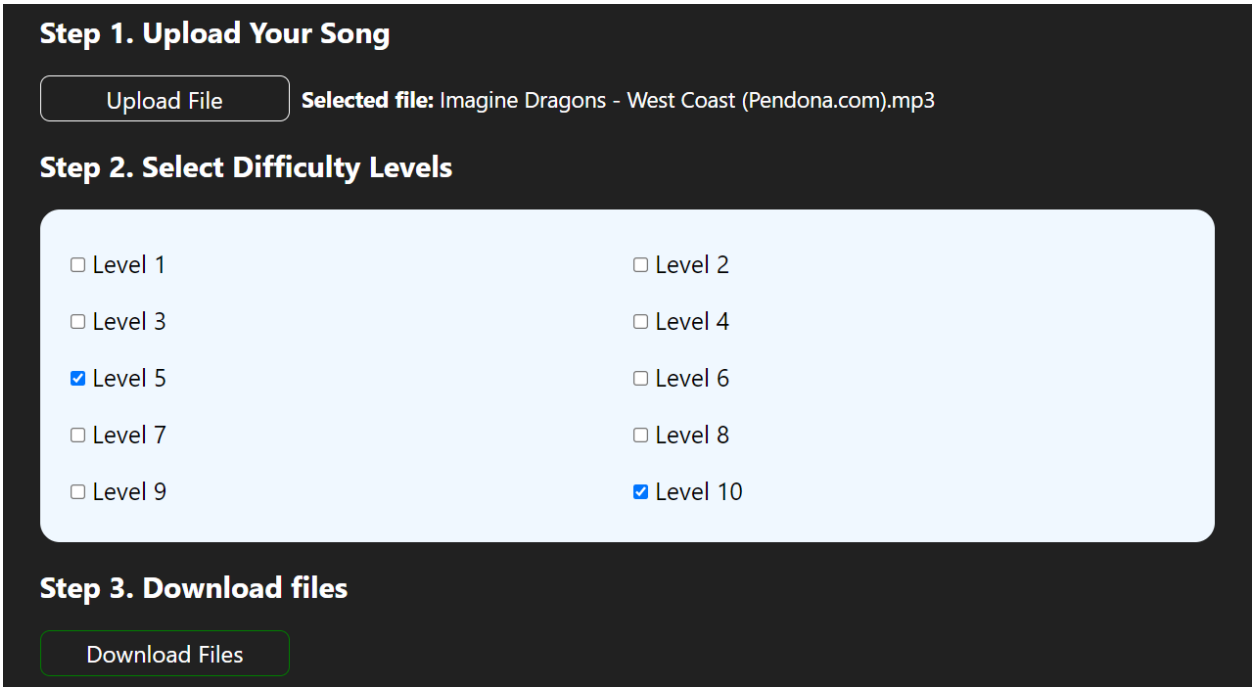

图 24：Web 应用程序部分 2

前端使用 JavaScript 和 React 创建。用户选择歌曲和难度等级后，前端会将歌曲文件和其他信息发送到后端。后端接收这些数据，并使用上述模型的评估模式生成新关卡。

后端使用以下处理流程：

1. 数据从前端接收 2. 使用“model.pth”、“all_characters.json”和“params.json”文件加载训练好的模型 3. 从“all_characters.json”中的所有唯一元素里随机选择起始元素 4. 根据关卡难度和歌曲时长选择预测长度；难度越高，预期的元素越多

5. 使用以下参数调用 evaluate 函数：模型、起始元素、预测长度和温度。该函数返回一个包含预测长度个元素的列表 6. 将列表解码为游戏可识别的 JSON 格式 7. 将结果发送回前端

后端使用 Python 和 FastAPI 构建，其处理流程如图25所示。

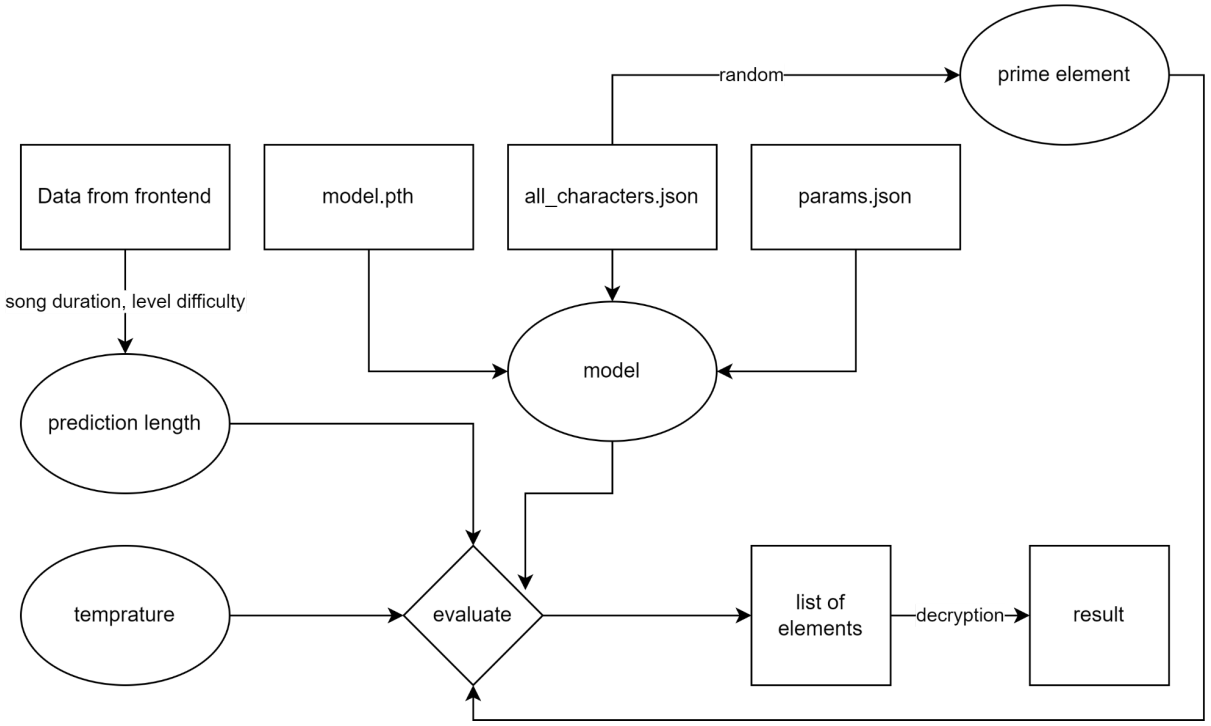

图25：后端流水线

5 结果

Ragnarock 这款 VR节奏游戏的节拍图创建（也称为编排学习）流程已实现自动化。该任务被分为三个部分：

1. 动作放置：确定节拍的时机。2. 动作选择：确定与选定节拍关联的符文应放置在空间中的什么位置。3. 应用程序创建：面向非技术用户的端到端应用程序。

数据（831 首歌曲）是从提供用户手动创建的自定义地图的网站解析得来的，并据此构建数据集。每个数据点解析了以下信息：歌曲编号、歌曲名称、歌曲、作者、歌曲文件名、每分钟节拍数，以及是否存在简单、普通、困难、专家或专家+难度。所有内容均保存到一个数据框中。共有 572 个简单关卡、130 个普通关卡、95 个困难关卡、147 个专家关卡和 186 个专家+关卡。

数据以字典格式读取，随后将这些字典中的元素替换为唯一标识符，并转换为列表，以便对每个关卡文件进一步处理数据。之后，数据便可用于模型。

对于动作放置的首项任务，使用 PLP 来确定节拍应放置的位置和数量。分析不同关卡中元素的分布数量后，选取任一难度关卡中的最大峰值数量作为最优值。

对于动作选择任务，使用了 915015 个元素进行模型训练。数据中有 15 个唯一元素。其中 4 个作为异常值被移除，另有 1 个作为冗余元素被移除。训练了一个隐藏层大小为 100 的单层门控循环单元（GRU）。模型及其他文件（“all_characters.json”和

“params.json”）已保存，以便后续进行模型重建并在 Web 应用程序中使用。

在应用程序创建的最后一项任务中，我们使用 JavaScript 和 React 构建了前端。用户可以在前端上传任意歌曲文件、选择不同的难度级别并下载新关卡。前端由使用 Python 和 FastAPI 构建的后端提供支持。后端接收歌曲文件和其他 JSON 格式的信息，使用自定义算法进行动作放置，加载训练好的模型进行动作选择，将所有内容保存到一个文件夹中，然后以 ZIP 文件夹的形式发送给用户。整个应用都托管在线。

参考文献

[1] https://www.ragnarock-vr.com/home

[2] https://beatsaber.com/

[3] DONAHUE, Chris；LIPTON, Zachary C.；MCAULEY, Julian. Dance Dance Convolution [在线]。2017年。可从 arXiv: 1703.06891 [cs, stat] 获取

[4] Luc, Ronald.（2020）。DeepSaber：用于高维编舞的深度学习。

[5] https://ragnacustoms.com

[6] Grosche, P., &Muller, M.（2011）。“从音乐录音中提取主导局部脉冲信息。”IEEE音频、语音与语言处理汇刊，19(6)，1688-1701。

[7] MOORE, Ben。《适用于 2019 [在线] [的最佳VR游戏》，访问日期：2020-02-22]。来源：https://www.pcmag.com/news/the-best-vr-games-for-2019。

[8] 吴世强、黄颖、管虎、张树武、刘杰，《一种基于扩频的音频水印方法：采用主导局部脉冲提取实现嵌入位置自适应》，2021年国际文化导向科学与技术会议（ICCST），第324–329页，2021年。

[9] Fikana Mahardika Cantri、Zaqiatud Darojah、Endah Suryawati Ningrum，《基于累积分数的实时音乐节拍检测系统》，2019年国际电子学研讨会（IES），第293–298页，2019年。

[10] Rumelhart、David E、Hinton、Geoffrey E 和 Williams、Ronald J（1985年9月）。通过误差传播学习内部表征。技术报告。

ICS 8504。加利福尼亚州圣迭戈：加利福尼亚大学认知科学研究所。

[11] Jordan, Michael I.（1986年5月）。序列顺序：一种并行分布式处理方法。技术报告 ICS 8604。加利福尼亚州圣迭戈：加利福尼亚大学认知科学研究所。

[12] R. Dey 和 F. M. Salem，《门控循环单元（GRU）神经网络的门变体》，2017年IEEE第60届中西部电路与系统国际研讨会（MWSCAS），2017年，第1597-1600页，doi：10.1109/MWSCAS.2017.8053243。

[13] J. Chung、C. Gulcehre、K. Cho 和 Y. Bengio，《门控循环神经网络在序列建模中的实证评估》，2014年。

[14] C. Paseddula 和 S. V. Gangashetty，《基于梅尔频率倒谱系数与逆梅尔频率倒谱系数分数融合的基于深度神经网络的声学场景分类》，2018年IEEE第13届工业与信息系统国际会议（ICIIS），2018年，第18-21页，doi：10.1109/ICIINFS.2018.8721379。

[15] HOCHREITER, Sepp；SCHMIDHUBER, Jürgen。《长短期记忆》。神经计算 [在线]。1997年，第9卷，第8期，第1735–1780页[访问日期：2020年7月04]。ISSN 0899-7667。DOI：10.1162/neco.1997.9.8.1735。

附录 A

学士学位论文程序实现的代码可通过以下 GitHub 链接找到：

https://github.com/MeriDK/Thesis