# 2023/3/31

训练数据应该来自BVH,且同一角色的训练数据应该严格一致(当前阶段).

BVH文件root是6个通道(位置旋转),joint是3个通道(旋转),旋转通道的顺序就是旋转顺序,为了避免问题,建议转换成四元数.

# 2023/3/20

这是一整套工具设计,按不同的难度目标分为两个阶段和多个步骤.最终目的是实现通用可定制的动画内容创建工具.

数据处理阶段

1. 以fbx为标准格式文件,从fbx种获取符合格式的资源.需要带有父子关系,位置,旋转的序列信息,甚至不需要时间信息.(初期可以临时用符合标准的文本数据)

|  |  |
| --- | --- |
| 动画补间阶段 |  |
| 1. 使用简单的单条骨骼动画数据,实现一组有父子关系的有序空间点,提供A(几帧空间和时间),生成一段连续的飘带动画,可选尽量保留原始pose的程度,支持可选首尾循环优化 如下图所示 2. 在1的基础上,添加碰撞点,提供A(几帧空间和时间)+B(提供碰撞点的逐帧空间和时间),生成一段合理的连续的飘带动画,物理参数取决于A的pose(空间和时间),开启碰撞时,如果A中原本就有不合理的穿插部分,可选忽略或强制纠正;添加碰撞后,仍然要能支持可选的首尾循环 3. 在2的基础上,添加约束,几组有序点可以作为一组类似网格点的方式计算,提供几组A(几帧空间和时间)和一组B(提供碰撞点的逐帧空间和时间,生成一段合理的连续的布料动画,并支持基于数据A的可变长度.带有约束后,仍要支持可选的首尾循环优化. |  |
| 1. 在以上的基础上,有序点支持体积计算,比如两个点之间的连接可以是任意宽高的长方体,并且至少支持平面,长方体,球体,胶囊体四种碰撞体,更好的支持碰撞和自转. |  |
| 1. 实现碰撞数据可以不按逐帧提供,可随A(几帧空间和时间)数据添加标记一同提供,并且能参与到过渡种,碰撞体之间不发生碰撞 |  |
| 1. 使用更多的数据,优化支持带有复杂分支的有序点的动画生成,比如植物枝杈,多头蛇 |  |
| 1. 使用更丰富的数据,优化支持生物骨架的动画生成支持 |  |
| 1. 可以标记哪些空间点(骨骼),属于生物,哪些属于物品,进行风格不同的计算 |  |
| 1. 对于已标记的生物角色,可以再标记某些骨骼可产生伪肌肉抖动效果及强度. |  |
|  |  |
| 动画产出阶段(基于以上成果) |  |
| 1. 使用新的动画数据,主要体现时间节奏差异,实现可以仅基于空间序列创建动画,可由程序生成时间间隔. |  |
| 1. 使用大量带详尽标记的对齐数据(风格,性别,性格,状态,特点,动作,速度,是什么帧(比如高位帧,极限帧,关键帧,叙事帧,缓冲帧)等等,数据要大,避免过拟合),实现基于给定的角色骨架(带有生物标记),用文本生成pose(一组有序点,一帧的空间数据),可选根据角色骨架生成物体pose(带有物体标记),这步结果可单独导出保存,便于手动优化重新导入 |  |
| 1. 根据提供的pose序列(只有顺序,没有时间),批量生成动画(时间间隔,过渡帧) 2. 可以设置遮罩标记,仅重新生成部分pose和动画. 3. 可通过少量对齐数据训练微调模型,缩小生成的内容范围. |  |

个人实现

1. 将1个pose描述为一个符号(文字),空位自动填充:比如:”p1......p2”,在p1和p2之间自动填充6个pose(每个点),再比如”...p1......p2...”,所有的点都自动填充.(我认为,这比较像利用RNN实现找规律填空…)

ChatGPT建议我学习Scikit-learn和Keras

对于初学者，Scikit-learn和Keras是比较容易上手的机器学习库。Scikit-learn提供了简单易用的接口和丰富的文档，适合初学者掌握基本的机器学习概念和算法。而Keras则是一个高层次的神经网络API，基于TensorFlow等底层框架，可以快速构建和训练神经网络模型。Keras也提供了简单易用的接口和丰富的文档，适合初学者掌握神经网络的基本知识。

相对来说，TensorFlow和PyTorch更适合那些有一定编程基础和机器学习知识的人。这两个库提供了更灵活的接口和更深入的控制能力，适合深入研究机器学习和深度学习领域。

Pandas则是一个数据处理和分析的库，适合对数据进行预处理和特征提取，但它并不是一个机器学习库，而是一个数据处理和分析工具。

总之，哪个库最简单和容易上手取决于你的学习目标和背景。如果你是完全的初学者，建议从Scikit-learn和Keras开始，逐步提高自己的知识水平。

ChatGPT建议我使用RNN循坏神经网络算法或者seq2seq模型(基于RNN网络模型构建),或者GAN生成对抗网络

算法推荐

对于需要通过大量连续数据训练生成连续数据的AI任务，建议使用循环神经网络（RNN）或其变体来完成。

RNN是一种神经网络结构，可以处理序列数据，其中每个输入都与前面的输入相关。RNN通过使用带有循环反馈的隐藏层来记忆先前的输入，并输出新的预测值。由于其循环性质和对序列数据的适应性，RNN在许多序列任务中取得了良好的效果，如语言模型、机器翻译、音频处理等。而在连续数据生成任务中，RNN可以通过训练输入数据序列和目标输出序列来生成新的连续数据序列，如音乐生成、图像生成等。

此外，还可以使用**长短时记忆网络（LSTM）或门控循环单元网络（GRU）**等RNN变体来处理更长时间序列数据，并且能够更好地控制记忆和遗忘的程度，从而取得更好的效果。

总之，对于需要生成连续数据的AI任务，RNN及其变体是一种很好的选择，它们能够从大量连续数据中学习到数据的模式，并生成新的连续数据。

无监督学习

RNN可以进行**无监督学习**。无监督学习是指在训练数据没有标签的情况下，让模型自动发现数据中的潜在结构和规律。在RNN中，可以使用自编码器（Autoencoder）等无监督学习算法来学习数据中的特征，并生成新的序列数据。

自编码器是一种将输入数据压缩为潜在表示，然后将其解码回原始输入的神经网络模型。对于序列数据，可以使用**循环自编码器**（Recurrent Autoencoder，RAE）来学习序列数据的潜在结构。RAE使用RNN编码器和解码器来学习输入序列的潜在表示，并将其解码回原始输入序列。

在无监督学习中，训练数据通常是不带标签的，因此需要定义一种自然的损失函数来度量模型的性能。在自编码器中，通常使用重构损失作为损失函数，即输入数据和解码后的输出数据之间的差异。通过最小化重构损失，模型可以学习到数据的潜在结构，并生成新的序列数据。

总之，RNN可以通过自编码器等无监督学习算法来进行无监督学习，学习序列数据的潜在结构和规律。

基本实现

RNN可以用于将输入的有序的3Dpose转换为符合输入的自然的3D动画。这个问题可以看作是一个序列到序列的问题，输入是一个有序的3Dpose序列，输出是一个符合输入的自然的3D动画序列。

具体来说，可以使用**序列到序列的RNN模型**，例如**基于LSTM或GRU的编码器-解码器模型**。编码器将输入的3Dpose序列编码成一个固定维度的潜在向量，解码器根据潜在向量以及先前的输出状态，生成符合输入的自然的3D动画序列。

为了训练这个模型，需要一些有标签的3D动画序列数据，可以使用监督学习的方法进行训练。将输入的3Dpose序列和对应的自然的3D动画序列作为训练样本，最小化预测3D动画序列和真实3D动画序列之间的差异，可以使用均方误差（MSE）或交叉熵等损失函数。

需要注意的是，**训练这个模型需要大量的标注数据和计算资源，而且还需要一些预处理和后处理的技巧来提高模型的性能。**因此，建议先学习一些基础的机器学习和深度学习知识，并熟悉一些相关的Python库，如TensorFlow、PyTorch、Keras等。

对于需要通过大量连续数据训练生成连续数据的AI任务，并且需要使用一些特征数据作为输入来生成输出，建议使用序列到序列模型或变体来完成。

序列到序列（seq2seq）模型是一种常见的序列学习框架，可以在给定输入序列的情况下，生成对应的输出序列。该模型通常由编码器和解码器两个部分组成，编码器将输入序列编码成一个固定长度的向量，解码器将此向量作为输入，并生成输出序列。

在生成连续数据的任务中，可以使用seq2seq模型，其中输入序列包括历史连续数据和一些特征数据，输出序列为未来连续数据。这样，模型可以在给定历史数据和特征的情况下，预测未来数据。

此外，还可以使用变体，如带注意力机制的seq2seq模型，可以更好地关注输入序列中重要的特征数据，并生成更准确的连续数据。另外，对于大量的连续数据，也可以使用一些预处理技术，如滑动窗口、差分等，来减少数据的复杂度和噪声。

总之，对于需要通过大量连续数据训练生成连续数据的AI任务，并且需要使用一些特征数据作为输入来生成输出，序列到序列模型或变体是一种很好的选择，它们可以从大量连续数据中学习到数据的模式，并生成新的连续数据。

对于需要从输入的几个3D Pose之间生成一段自然的3D动画的AI任务，可以考虑使用生成对抗网络（GAN）或变体来完成。

GAN是一种神经网络结构，由两个子网络组成：生成器和判别器。生成器将一个随机噪声向量映射到一个新的数据点，而判别器则评估该数据点是否与训练数据相似。这两个子网络在训练过程中相互对抗，最终生成器可以生成与真实数据相似的数据点。

在生成3D动画的任务中，可以将多个3D Pose作为输入，生成器可以将这些3D Pose转换为一段自然的3D动画。判别器可以评估生成的3D动画是否与真实3D动画相似。通过不断地迭代训练，生成器可以学习到从多个3D Pose中生成自然的3D动画，而判别器则可以评估生成的3D动画的真实性。

此外，还可以使用一些变体，如条件GAN（cGAN），将额外的信息作为条件输入，以控制生成的3D动画的特定属性。比如，可以将音乐的节奏、情感等作为条件输入，以生成与音乐节奏和情感相匹配的3D动画。

总之，对于从输入的几个3D Pose之间生成一段自然的3D动画的AI任务，GAN及其变体是一种很好的选择。它们可以学习到从输入数据中生成符合真实数据分布的新数据，从而生成自然的3D动画。

通过大量文本数据,实现生成字与字间的填空的AI,用什么框架和算法比较好

这个问题涉及到自然语言处理（Natural Language Processing，简称NLP）领域中的文本生成任务，可以采用循环神经网络（Recurrent Neural Networks，简称RNN）及其变种来解决。

在深度学习框架方面，可以选择TensorFlow、PyTorch等，它们都提供了丰富的深度学习算法和工具，方便进行模型训练和推理。

至于具体的算法选择，可以考虑使用基于RNN的语言模型，如循环神经网络语言模型（Recurrent Neural Network Language Model，简称RNNLM）、长短时记忆网络语言模型（Long Short-Term Memory Language Model，简称LSTM LM）等。这些模型可以通过学习语言中单词之间的概率分布来实现文本生成任务。

在填空文本生成任务中，可以采用序列到序列（Sequence-to-Sequence，简称Seq2Seq）模型，其包含编码器和解码器两部分。编码器将输入文本序列映射到一个固定长度的向量表示，然后解码器将该向量作为起点生成新的文本序列。这种方法可以使模型更好地理解输入文本的含义，并生成与输入文本上下文相关的文本输出。

在实现过程中，还需要考虑到数据集的选择和预处理，以及超参数的调整等问题。

那如果把这些"字"作为矩阵,实现矩阵与矩阵之间的填空,用什么框架和算法比较好

如果将这些字作为矩阵，则可以使用深度学习模型来解决矩阵填空的问题。一种比较常用的深度学习模型是基于神经网络的自编码器（autoencoder）。自编码器是一种无监督学习算法，可以从未标记的数据中学习输入数据的压缩表示。其基本思想是通过将输入数据压缩成低维向量，再将低维向量还原成输入数据，从而使得网络可以学习到输入数据的内在特征。在填空问题中，我们可以将输入矩阵中的一些元素随机遮盖住，然后将其作为输入，让自编码器来预测被遮盖住的元素。当自编码器的预测误差足够小时，我们就可以得到一个填好空的矩阵。

在实现自编码器时，可以使用许多深度学习框架，如TensorFlow、Keras、PyTorch等。其中，PyTorch是一个灵活性高、易于使用的框架，可以快速地实现深度学习模型。具体来说，可以使用PyTorch中的nn模块来定义自编码器的网络结构，并使用optim模块来定义优化器。在训练过程中，可以使用PyTorch提供的autograd机制来自动计算网络中的梯度，并使用PyTorch的backward方法来进行反向传播。

通过大量文本数据,实现生成多个字与字之间的填空,连成通顺的句子的AI,用什么框架和算法比较好?如果把这些"字"作为矩阵,实现矩阵与矩阵之间的填空,并且产生基于训练数据的比较自然的矩阵数值变化,用什么框架和算法比较好?

针对第一个问题，生成字与字间的填空可以使用基于自然语言处理的技术，比如语言模型（Language Model）或者序列到序列模型（Sequence-to-Sequence Model），通常采用深度学习框架来实现，比较常用的包括TensorFlow、PyTorch等。

对于第二个问题，如果将这些“字”作为矩阵，可以考虑采用基于矩阵分解的方法，比如奇异值分解（Singular Value Decomposition，简称SVD）或者主成分分析（Principal Component Analysis，简称PCA）。这些方法可以帮助我们从数据中提取出重要的特征，并且能够很好地处理缺失数据和噪声。同样地，也可以采用深度学习框架来实现，比如使用循环神经网络（Recurrent Neural Network，简称RNN）或者卷积神经网络（Convolutional Neural Network，简称CNN）来学习数据中的模式，并且生成自然的矩阵数值变化。

总的来说，选择哪种算法和框架还要根据具体的应用场景和需求来确定，需要对各种方法进行比较和评估，找到最适合自己的方法。