tensorflow.data.Dataset.interleave(

map\_fn, cycle\_length,block\_length, num\_parallel\_calls=tf.data.AUTOTUNE)

先按照cycle\_length从dataset取数，对取出的数应用map\_fn生成临时dataset, 再依次从临时数据集取block\_length个数据生成返还的数据集

*Eg:*

*import tensorflow as tf*

*# 创建基础数据集*

*dataset = tf.data.Dataset.range(10)*

*# 定义映射函数：为每个数字创建一个包含重复数字的数据集*

*def map\_fn(x):*

*return tf.data.Dataset.from\_tensors(x).repeat(3)*

*# 使用 interleave*

*interleaved\_dataset = dataset.interleave(*

*map\_fn,*

*cycle\_length=2, # 同时处理2个数据集*

*block\_length=1 # 每次从一个数据集中取1个元素*

*)*

*# 查看结果*

*for item in interleaved\_dataset:*

*print(item.numpy(), end=' ')*

*# 输出可能是：0 1 0 1 0 1 2 3 2 3 2 3 4 5 4 5 4 5 ...*

TFRecord相关：

*import tensorflow as tf*

*# 1. 首先看如何写入 TFRecord*

*# 辅助函数，用于创建特征*

*def \_float\_feature(values):*

*return tf.train.Feature(float\_list=tf.train.FloatList(value=values))*

*# 创建一个样本数据*

*ppg\_data = [0.1, 0.2, 0.3] \* 58 + [0.4] # 175个值*

*label\_data = [1.0, 0.0] # 二分类标签*

*# 将数据转换为 TFRecord 格式*

*feature = { 'ppg': \_float\_feature(ppg\_data), 'label': \_float\_feature(label\_data) }*

*# 创建 Example*

*example = tf.train.Example(features=tf.train.Features(feature=feature))*

*# 序列化 Example*

*serialized\_example = example.SerializeToString()*

*# 2. 然后看如何读取和解析 TFRecord*

*# 定义特征格式*

feature\_description = { 'ppg': tf.io.FixedLenFeature([175], tf.float32), 'label': tf.io.FixedLenFeature([2], tf.float32) }

*# 解析单个样本*

*parsed\_features = tf.io.parse\_single\_example(*

*serialized\_example, # 序列化的样本*

*feature\_description # 特征格式描述*

*)*

*# 现在 parsed\_features 是一个字典，包含解析后的数据*

*ppg = parsed\_features['ppg'] # 形状为 [175] 的张量*

*label = parsed\_features['label'] # 形状为 [2] 的张量*

numpy.squeeze()：

这个函数用于删除数组中长度为1的轴（维度）。它可以帮助降低数组的维度。

*import numpy as np*

*# 创建一个形状为 (1, 3) 的数组*

*a = np.zeros((1, 3, 1, 4)) #[[0,0,0]]*

*# 使用squeeze()删除长度为1的轴*

*b = np.squeeze(a) #[0,0,0]*

*print("原始数组形状:", a.shape) # (1, 3, 1, 4)*

*print("squeeze后的形状:", b.shape) # (3, 4)*

numpy.cumsum():

累积求和函数，它会计算数组中元素的累积和。默认情况下，它会沿着最后一个轴进行累积求和。

*# 一维数组的累积和*

*a = np.array([1, 2, 3, 4])*

*print("一维数组累积和:", np.cumsum(a)) # [1 3 6 10]*

*# 二维数组的累积和*

*b = np.array([[1, 2, 3], [4, 5, 6]])*

*print("默认轴累积和:\n", np.cumsum(b)) #[1,3,6,10,15,21]*

*print("指定轴累积和:\n", np.cumsum(b, axis=0)) #[[1,2,3][5,7,9]]*

numpy.unique():

返回数组中的唯一值，并可以选择是否返回索引、计数等附加信息。

*import numpy as np*

*# 基本用法*

*a = np.array([1, 2, 2, 3, 3, 3, 4, 4, 5])*

*print("唯一值:", np.unique(a)) #[1,2,3,4,5]*

*# 返回更多信息*

*unique\_values, indices, counts = np.unique(a, return\_index=True, return\_counts=True)*

*print("唯一值:", unique\_values) #[1, 2, 3, 4, 5]*

*print("首次出现的索引:", indices) #[0, 1, 3, 6, 8]*

*print("每个唯一值的计数:", counts) #[1, 2, 3, 2, 1]*