Dataset分析

class Data(OldData):

这段代码定义了一个自定义的数据类 Data，它继承自 OldData 类。在这个自定义的 Data 类中，重写了 \_\_inc\_\_ 方法。在进行批处理时，可以根据节点的数量自动调整边的索引，以确保正确的连接

Normalize by a constant coefficient——按常数系数归一化

class Normalize(object):

这段代码定义了一个名为 **Normalize** 的类，用于对数据进行归一化操作

Map glove data to inspire hand data

def linear\_map(x\_, min\_, max\_, min\_hat, max\_hat):

该函数实现了线性映射的功能，将一个区间内的数值映射到另一个区间内。

def map\_glove\_to\_inspire\_hand(glove\_angles):

该函数将Wiseglove手套的角度测量值线性映射到Inspire手的关节角度。

Parse H5 File

def parse\_h5(filename, selected\_key=None):

解析H5文件并提取其中的手部数据。函数的输入参数包括H5文件的文件名（filename）以及可选的选择键（selected\_key），用于指定要提取的数据集。函数首先打开指定的H5文件，然后遍历其中的数据集（或者指定的数据集），从中提取左右手部的角度信息、位置信息和四元数信息。

parse\_h5函数的作用是解析H5文件中的手部数据，并将其转换为PyTorch的Data对象，以便用于训练深度学习模型。

def parse\_arm(l\_shoulder\_euler, l\_elbow\_euler, l\_wrist\_euler, r\_shoulder\_euler, r\_elbow\_euler, r\_wrist\_euler,  
 l\_shoulder\_pos, l\_elbow\_pos, l\_wrist\_pos, r\_shoulder\_pos, r\_elbow\_pos, r\_wrist\_pos,  
 l\_shoulder\_quat, l\_elbow\_quat, l\_wrist\_quat, r\_shoulder\_quat, r\_elbow\_quat, r\_wrist\_quat):

这段代码定义了一个名为parse\_arm的函数，其作用是根据给定的关节欧拉角、关节位置、关节四元数等信息解析并构造手臂的数据对象。将给定的手臂关节信息解析并转换为PyTorch的Data对象，以便用于训练深度学习模型。

在上述代码中，parse\_h5函数负责解析H5文件并提取手部数据，而parse\_arm函数负责根据提取的数据构造手部的Data对象。这两个函数之间的联系在于parse\_h5函数在循环遍历每个帧时调用了parse\_arm函数来构造手部的Data对象。

具体来说，parse\_h5函数首先从H5文件中读取了左右手部的角度信息、位置信息和四元数信息等数据，然后对每个帧进行循环遍历，在循环中调用了parse\_arm函数来构造手部的Data对象。因此，parse\_arm函数在parse\_h5函数中扮演了构造手部数据的角色，二者共同完成了从H5文件中提取手部数据并构造Data对象的任务

Source Dataset for Sign Language

class SignDataset(InMemoryDataset):

def \_\_init\_\_(self, root, transform=None, pre\_transform=None):

构造函数初始化SignDataset对象。它接受root（数据集根目录）、transform（数据转换操作）和pre\_transform（预处理转换操作）作为参数，并调用父类的构造函数初始化对象。

def raw\_file\_names(self):

获取原始数据文件的完整路径列表

def processed\_file\_names(self):

定义了处理后数据文件的名称列表。在这种情况下，只返回包含一个元素'data.pt'的列表，表明处理后的数据将存储为名为'data.pt'的文件。

def process(self):  
 process方法用于处理原始数据并将其转换为PyTorch张量格式。它首先遍历所有的原始数据 文件，使用parse\_h5函数解析每个文件中的数据，并将解析后的数据存储在data\_list列表中。然后，如果提供了pre\_filter函数，它将对数据进行过滤。接着，如果提供了pre\_transform函数，它将对数据进行预处理转换。最后，它调用self.collate函数对数据进行整理，并将整理后的数据与对应的切片信息一起保存为一个元组，使用torch.save函数将元组保存到processed\_paths列表中的第一个文件中

实现了一个用于处理手语数据集的PyTorch数据集类，负责加载、解析、过滤和预处理原始数据，并将其转换为PyTorch张量格式以便用于训练模型。

parse h5 with hand

def parse\_h5\_hand(filename, selected\_key=None):

def parse\_glove\_pos(glove\_pos):

Source Dataset for Sign Language with Hand

class SignOnlyHand(InMemoryDataset):

内容同上

parse h5 with all data

def parse\_all(filename, selected\_key=None):

Source Dataset for Sign Language with Hand

class SignAll(InMemoryDataset):

Target Dataset for Yumi Arm

class Yumi(InMemoryDataset):

yumi\_cfg =

包含Yumi机械臂关节名称、边缘连接、根节点名称、末端执行器等信息的字典

def \_\_init\_\_(self, root, transform=None, pre\_transform=None):

初始化方法，用于创建Yumi类的实例。在初始化过程中加载并处理数据集。

def raw\_file\_names(self):

获取数据集中原始文件的文件名列表，这些文件是以.urdf为扩展名的文件。

def processed\_file\_names(self):

获取处理后的数据文件名列表

def process(self):

遍历原始文件列表，对每个文件调用yumi2graph函数进行转换。

对转换后的数据进行预处理和转换，然后将其添加到data\_list列表中。

最后，将所有数据整理为PyTorch的Data对象，并保存到文件中。

以上代码定义了一个名为Yumi的类，该类是PyTorch的InMemoryDataset的子类，用于表示Yumi机械臂的数据集。该类的主要作用是加载和处理Yumi机械臂的数据，并将其转换为PyTorch可用的数据格式。Yumi类用于加载和处理Yumi机械臂的数据集，将其转换为PyTorch中的数据对象，以便进行后续的机器学习模型训练和评估。

Target Dataset for Inspire Hand

class InspireHand(InMemoryDataset):

Target Dataset for Yumi with Inspire Hand

class YumiInspire(InMemoryDataset):

主函数：

if \_\_name\_\_ == '\_\_main\_\_':

代码模块化具体架构

# 训练（main.py—>dataset.py—>urdf2graph.py）

train\_set=getattr(dataset,cfg.DATASET.TRAIN.SOURCE\_NAME)(root=cfg.DATASET.TRAIN.

SOURCE\_PATH, pre\_transform=pre\_transform)

**（train\_set** 实际上是 **dataset.py** 中某个数据集类的一个实例化对象。）

——>dataset.py主函数：

if \_\_name\_\_ == '\_\_main\_\_':

yumi\_dataset = Yumi(root='./data/target/yumi')

——>Yumi类

class Yumi(InMemoryDataset):

…

data\_list.append(yumi2graph(file, self.yumi\_cfg))

…

——> yumi2graph函数：

def yumi2graph(urdf\_file, cfg):

该函数的主要作用如下：

加载Yumi机械臂的URDF文件，并解析其中的关节参数。

根据提供的配置信息，构建机械臂的拓扑结构，包括节点、边缘索引和边缘特征等。

计算节点之间的相对位置信息，包括节点到根节点、肩部和肘部的距离等。

构建图形表示的数据结构，包括节点特征、边缘索引、边缘特征、节点掩码等，并返回该数据结构。

函数内部实现细节如下：

加载URDF文件并解析其中的关节参数，存储在名为joints的字典中。

根据提供的配置信息构建机械臂的拓扑结构，包括节点名称、边缘连接等。

计算节点之间的相对位置信息，包括节点到根节点、肩部和肘部的距离等。

构建图形表示的数据结构，包括节点特征、边缘索引、边缘特征、节点掩码等。

使用前向运动学模型计算机械臂各个关节的位置，并可视化结果。

最后，函数返回构建好的图形表示的数据结构，该数据结构可以用于机器学习模型的训练和评估。

model = getattr(model, cfg.MODEL.NAME)().to(device)

——>model.py中的 cfg.MODEL.NAME类

class ArmNet(torch.nn.Module):

# 测试（inference.py—>dataset.py—>urdf2graph.py）

test\_data=parse\_all(filename=cfg.INFERENCE.MOTION.SOURCE,selected\_key=cfg.INFERENCE.

MOTION.KEY)

——> parse\_all(函数)

def parse\_all(filename, selected\_key=None):

model = getattr(model, cfg.MODEL.NAME)().to(device)

——>model.py中的 cfg.MODEL.NAME类

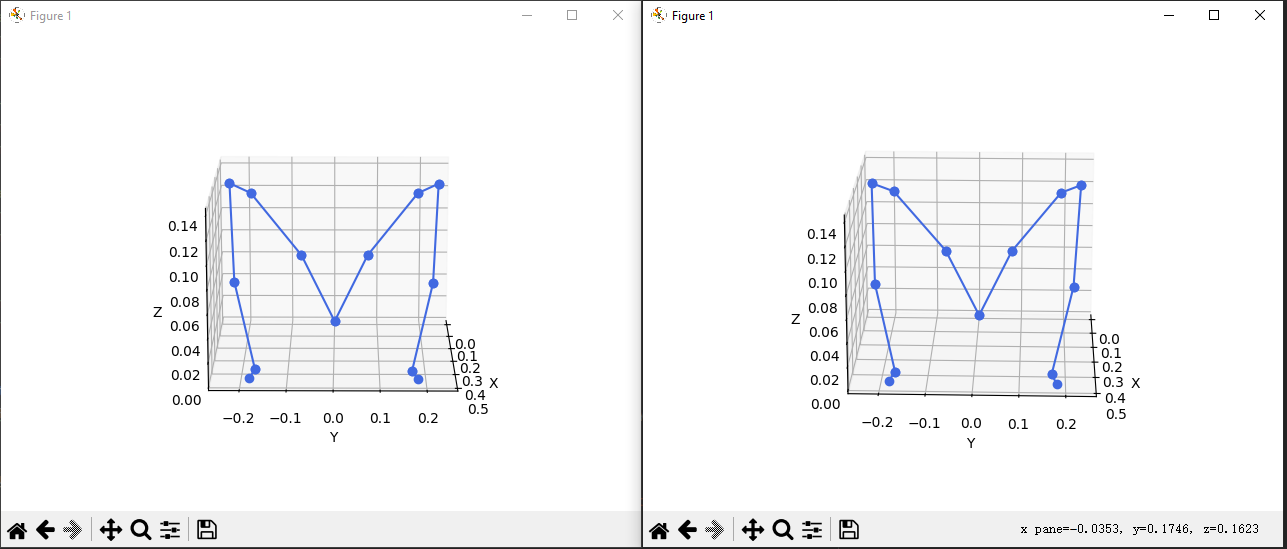
class ArmNet(torch.nn.Module):

其他函数的调用（如前向运动学kinematics.py，train.py,test.py）基本不变

手臂部分的处理对比

# 训练部分（main.py—>dataset.py—>urdf2graph.py）

代码完全一样

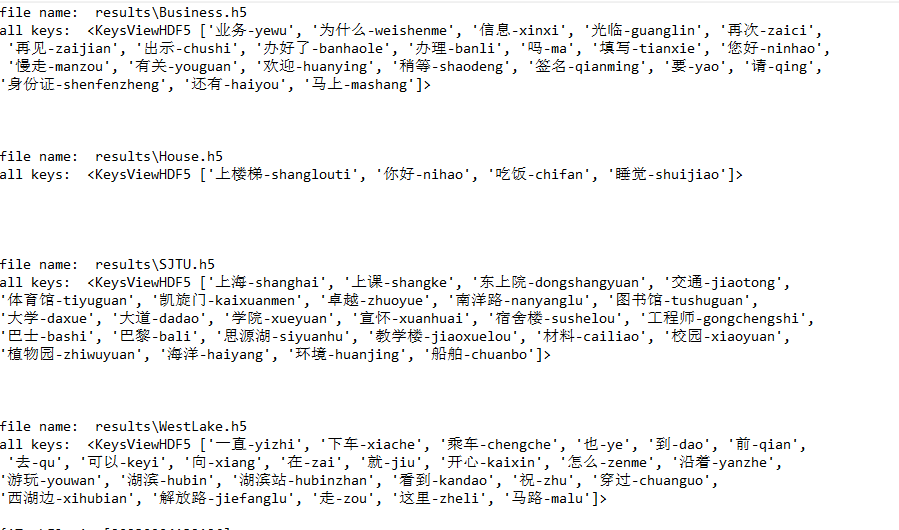
效果也完全一样

# 测试（inference.py—>dataset.py—>urdf2graph.py）

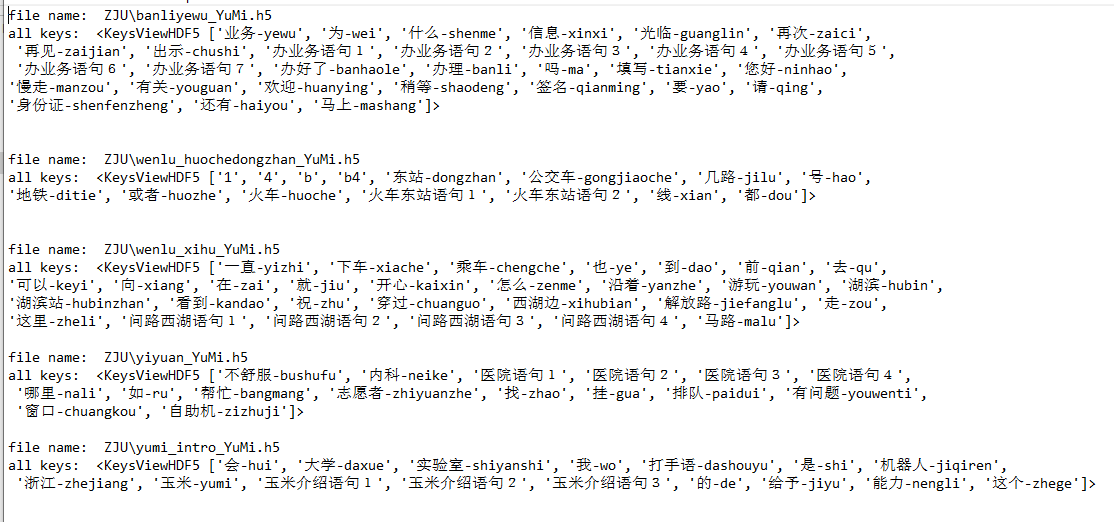
完全一样

包括model.py也是

HW数据集



ZJU数据集



运行结果

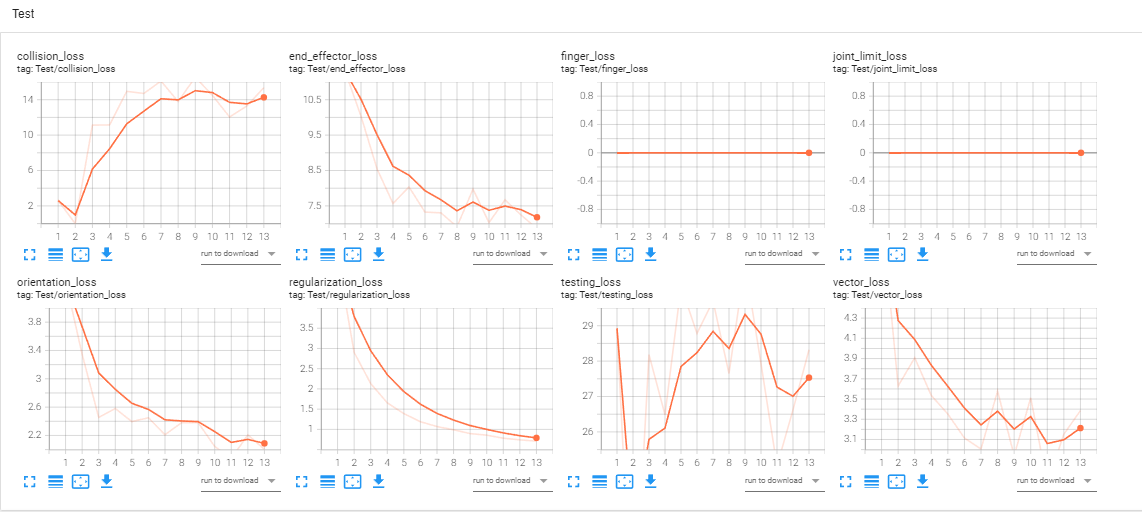
# ZJU数据+ZJU数据处理（环境测试）

## 训练：

数据：2+3同浙大完整

结果：

A screenshot of a graph

Description automatically generated

## 测试：

实际参考模型：best\_model\_epoch\_0001.pth

理论最优参考模型：best\_model\_epoch\_0550.pth

测试词汇：‘我-wo’

结果：

可以在yumi-gym仿真环境中运行，且运行效果和参考模型差不多，因为训练时间短效果没有那么好，但大致的轨迹姿态是差不多的，可以判断为项目成功运行。

# ZJU数据+ZJU数据处理（对照实验）

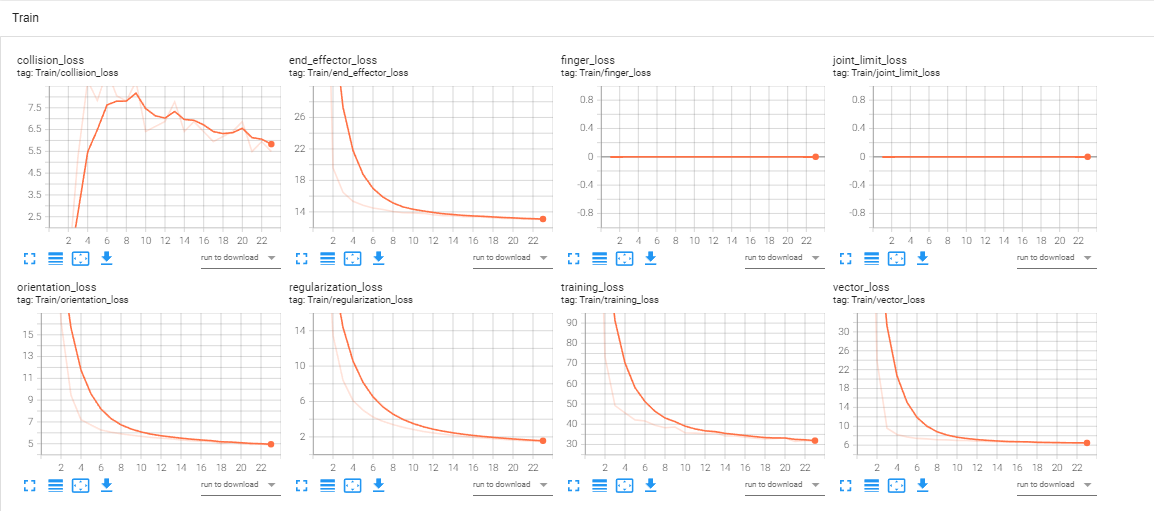
## 训练：

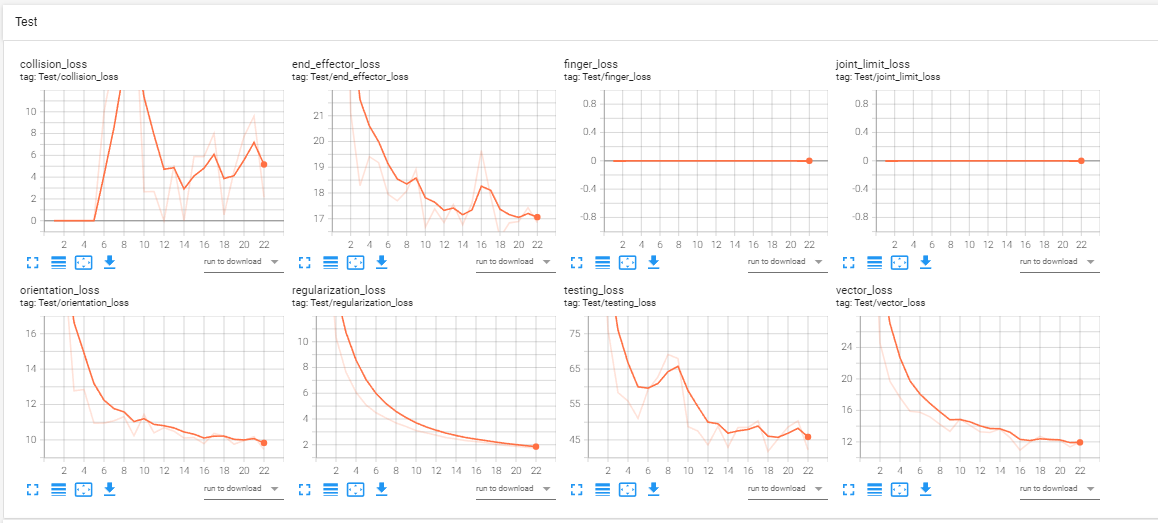
数据：

训练集：wenlu\_xihu\_YuMi.h5

测试集：banliyewu\_YuMi.h5

结果：





## 测试：

实际参考模型：best\_model\_epoch\_0017.pth

结果：

可以在yumi-gym仿真环境中运行，因为训练时间短效果没有那么好，但大致能动，大方向上没错，可以判断为项目成功运行。

# HW数据+HW数据处理（对照实验）

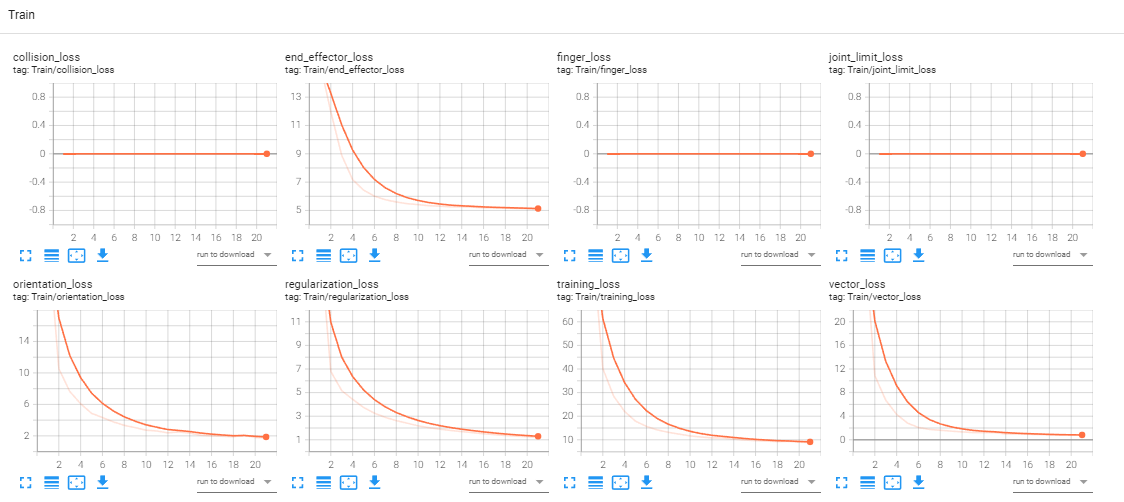
## 训练：

数据：

训练集：WestLake.h5

测试集：Business.h5

结果：



A screenshot of a graph

Description automatically generated

## 测试：

实际参考模型： /train/2024-03-07\_15-38-56/best\_model\_epoch\_000024.pth

结果：

错误，手还是往背后拧。

# ZJU数据+HW数据处理（对照实验）

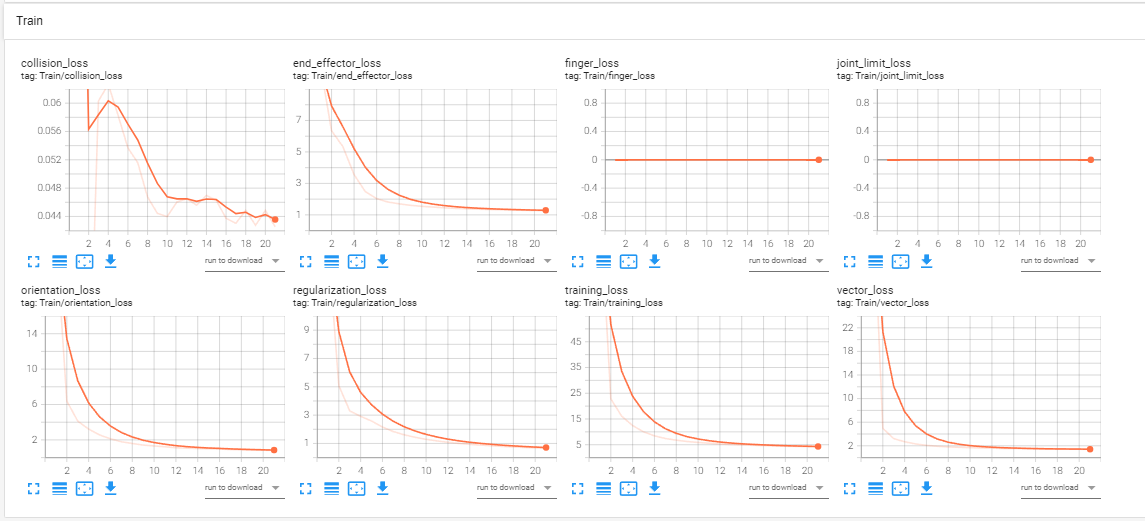
## 训练：

数据：

训练集：WestLake.h5

测试集：Business.h5

结果：



A graph of a function

Description automatically generated with medium confidence

## 测试：

实际参考模型：" /train/2024-03-07\_16-26-13/best\_model\_epoch\_000016.pth"

结果：正常运行，效果同ZJU数据+ZJU处理

# HW数据+ZJU数据处理（对照实验）

1

[[ 0.05047129 -0.00313617 0.73715753]

[ 0.05037737 -0.00325547 0.73715468]

[ 0.04984087 -0.00361918 0.73714867]

[ 0.0497219 -0.00379256 0.73714336]

[ 0.0496278 -0.00397186 0.73714176]

[ 0.04947054 -0.0041516 0.73713481]

[[ 0.05012046 -0.00368081 0.89971237]

[ 0.04998833 -0.00389955 0.89971007]

[ 0.0492078 -0.00457348 0.89970142]

[ 0.04897231 -0.00488675 0.89969775]

[ 0.04884011 -0.00521551 0.8996959 ]

[ 0.04854652 -0.00554595 0.89969035]

[ 0.04822779 -0.00587311 0.89968847]

[ 0.04813144 -0.00620072 0.89968771]

l\_shoulder\_pos: [[ 0.02338852 0.24312832 -0.00463492]

[ 0.02301429 0.24338367 -0.00463682]

[ 0.02269928 0.24363496 -0.00464433]

[ 0.02231883 0.24391481 -0.0046578 ]

[ 0.02196629 0.24423083 -0.00466603]

[ 0.02170469 0.24444648 -0.00472468]

[ 0.02142398 0.24468931 -0.00475913]

[ 0.02102534 0.24489912 -0.00481838]

[ 0.0207658 0.24506374 -0.00484794]

l\_elbow\_pos: [[ 0.05298389 0.26623651 -0.21064347]

[ 0.05266501 0.26642152 -0.21081537]

[ 0.05241583 0.26663484 -0.21080428]

[ 0.05215286 0.26685657 -0.21070248]

[ 0.05188815 0.26705935 -0.21071607]

[ 0.05173706 0.26718422 -0.21074516]

[ 0.05159338 0.26734581 -0.21076709]

l\_wrist\_pos: [[ 0.1136663 0.22415145 -0.43992913]

[ 0.11329892 0.22437575 -0.43999571]

[ 0.1131704 0.22448393 -0.43985885]

[ 0.11303596 0.22459199 -0.43986863]

[ 0.11298761 0.22467908 -0.43980747]

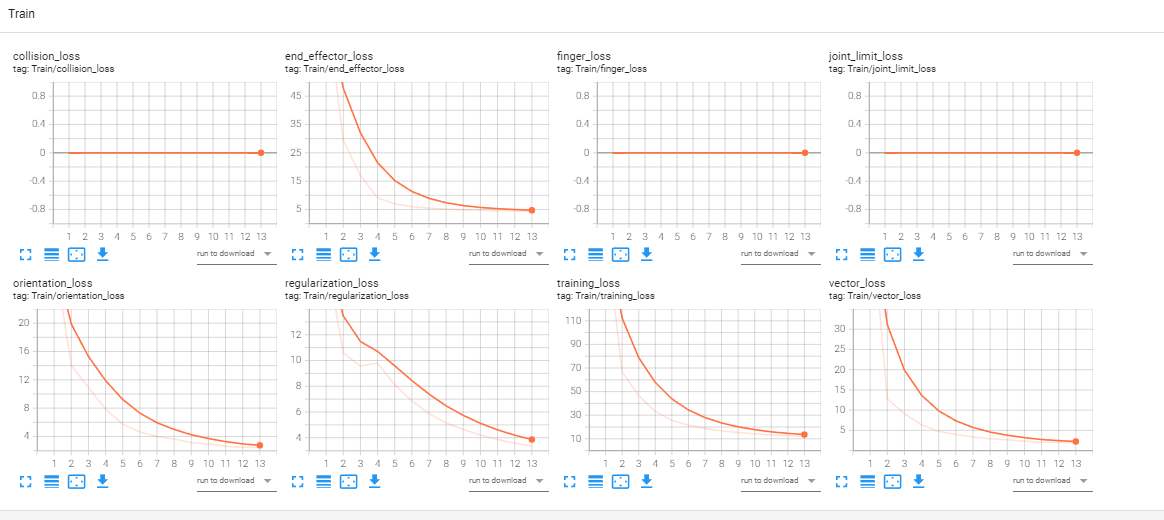
[ 0.11294407 0.2246988 -0.43978083]

[ 0.11288203 0.22477045 -0.43977427]

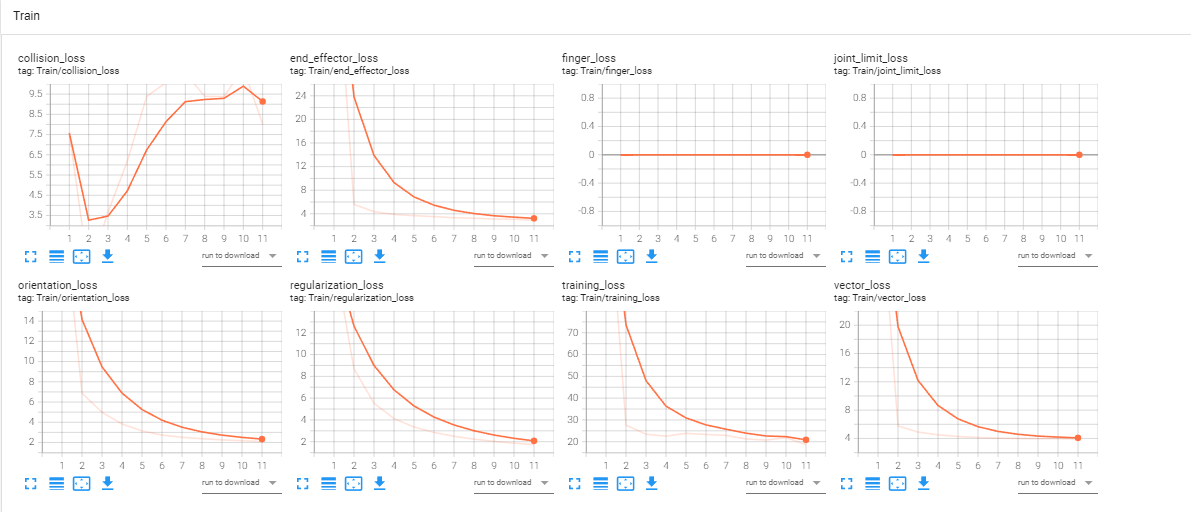
[ 0.11279042 0.2248167 -0.43979508]

[ 0.11270988 0.22484662 -0.439803 ]

[ 0.1126783 0.22494861 -0.43973976]

A graph of a function

Description automatically generated with medium confidence



A graph of a function

Description automatically generated with medium confidence