# ENNRF: Ensemble Feedforward Neural Network and Stable Random Forest for Pedestrian Dead Reckoning

**赵思衡　王科　林滋芊**
人工智能学院
南京大学
{zhaosh,scik,linzq}@smail.nju.edu.cn

**Abstract**

行人航迹检测 (PDR) 是一种常用的室内定位方法。在本次实验报告中，我们简要回顾了行人航迹检测的发展历史与现有算法，分析了其优势与缺点；介绍了我们收集的数据以及找到的公开数据集；介绍并实现了一种基于深度残差网络的 SOTA PDR 算法，并额外设计了迁移学习的模块以增强该方法的泛化性能；在测试数据上进行了实验并比较了各种传统算法以及其他神经网络学习算法的性能，并选择了集成神经网络与随机森林的方法作为最终提交方法；最后，介绍了可执行代码的架构、运行方法以及小组分工。

## 1 Introduction

在户外情况下，一个使用智能手机的行人的定位通常是使用智能手机内嵌的全球卫星导航系统 (GNSS) 接收器来完成。然而，当行人进入室内后，GNSS 信号通常是不可获得或者不准确的，因此此时的定位只能依靠于其他方法，比如基于视觉 [?]、无线电频率信号 [?] 或者惯性传感器 [?] 的方法。然而，基于视觉的方法需要在行人的身前装置一个深度敏感的 RGB 摄像头，基于无线电频率信号的方法需要额外装置一个电磁波发射器，这两种方法的额外要求都限制了其使用范围。而基于惯性传感器的方法，即行人航位推算 (PDR) [?]，只需使用绝大部分行人都配备的智能手机上的惯性传感装置即可，体现了该方法的广适用性。

一个惯性测量单元 (IMU) 由加速度计、陀螺仪和磁力计组成。在传统的 PDR 算法中，行人航迹的推断通常被解耦成四个部分：步态检测、步长估计、朝向检测、定位计算。步态使用加速度计进行检测；由于人步长的重复和固定性，步长可以根据经验或者生物力学进行推断 [?, ?, ?]；朝向检测通过对陀螺仪和磁力计进行积分可以进行估计

[?, ?]；最终，根据输入的初始状态和输出的步长、方向进行积分，可以计算出下一时间步的行人定位，具体流程如图 ?? 所示。然而在实践中，这种两次积分的方法会导致传感器偏差逐步积累，直至导致航迹估计的误差达到无法忽略的地步。

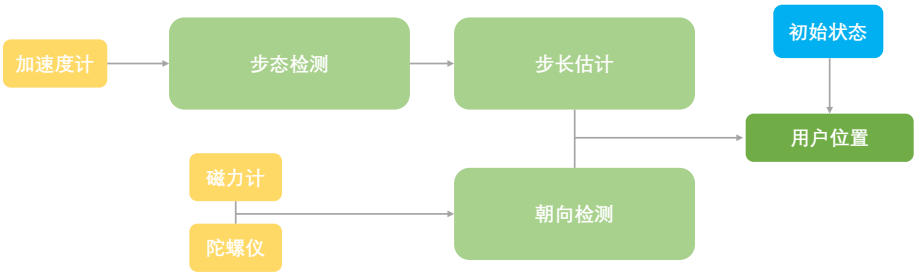


图 1: 传统步态检测算法的流程图：黄色模块表示输入，浅绿色模块表示处理系统，绿色模块表示输出，蓝色模块表示初始状态。

随着深度学习在控制 [?]、语音文字 [?]、图像 [?] 等各个领域击败了传统方法，达到了新的 SOTA。为了进一步提升性能，行人航迹检测也逐渐放弃了传统的卡尔曼滤波器 [?]，迎来了它的 Learning-based 时代。我们可以大致将这些工作分为两类：

**使用深度神经网络拟合速度向量，之后再进行积分来估计行人的位置**。这类工作的代表作是 RIDI [?]。RIDI 使用向量回归模型在智能手机坐标系中对速度进行回归，同时依靠传统的传感器混合方法来估计设备方向。回归拟合得到的速度用于在积分之前校正加速度计测量值。

**直接使用深度神经网络拟合行人位置和朝向**。第一篇 Deep-Learning based 的行人航迹推断方法 IONet [?] 使用 LSTM 网络，将归一化的加速度计和陀螺仪测量值作为输入来拟合一个预先定义好的时间窗口的位置与方向的变化值。然而，使用归一化的输入值会导致精确度的损失。后来的 RoNIN [?] 这篇工作成功使用了 ResNet 作为拟合网络，并提出了一种不受手机设备方向影响的架构。在这两篇工作的基础上，? 进一步使用 ResNet ，将神经网络模块化，提出了一种更加简便的方法，达到了新的 SOTA。

我们虽然也尝试过使用 ? 中的神经网络架构，并将其调整到我们的应用场景下，但是最终发现由于我们收集到的数据量太少，不适应于深度学习的场景，网络明显欠拟合，最终我们选择使用了适合于小样本学习的多层前馈神经网络和随机森林分别预测经纬度与方位角，作为最终提交方案。

## 2 Dataset

在本次实验中，我们不仅自己收集了数据 (??)，还使用了网上开源的公开数据集 (??) 用于模型的迁移学习。

### 2.1 Our Dataset

#### 2.1.1 Data Collection

我们联合其他小组使用 Pyphox 软件，使用多个设备进行为期一周的数据收集。实验设置包含加速度计，线加速度计，GPS 定位器，陀螺仪，磁力计，压力传感器六个传感器。每条数据的时间均大于 2 分钟，且根据不同的设备状态分为拿在手上、放在背包里、放在裤兜里等类别。该数据集分为 56 段，记录了总长 300 分钟的数据，其中传感器频率为 50Hz，GPS 定位器为 1Hz。我们将其中的 3 条用于训练的验证集，3 条用于训练的测试集。

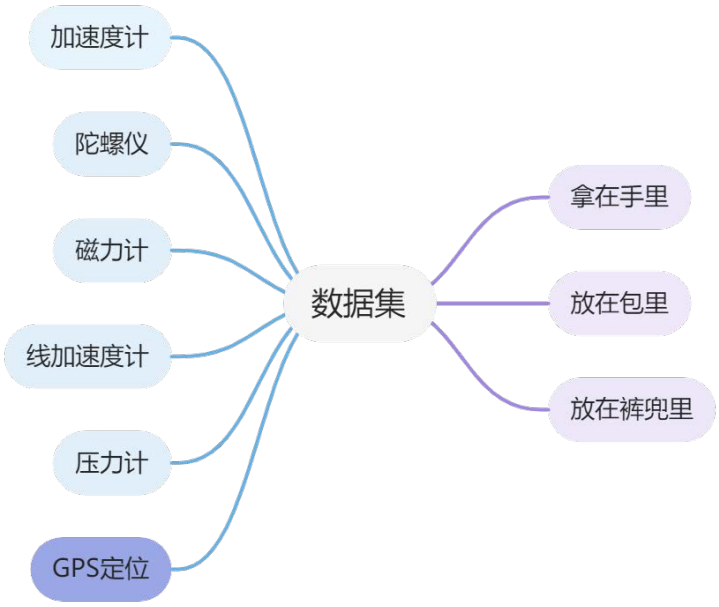


图 2: 数据集示例图：浅蓝色为传感器属性，深蓝色为 GPS 定位数据，粉色为收集到的不同场景的数据。

#### 2.1.2 Data Preprocessing

我们的数据来源于多个设备，在数据收集初步阶段数据量较少时 (此时仅包含 iPhone 测量的数据)，我们发现除去压力计外的传感器的时间戳是相同的，于是我们打算针对每一个 GPS 数据的时间戳寻找最相近的传感器时间戳进行数据对齐。然而加入安卓设备后，我们发现安卓设备的传感器时间戳是各不相同的，若是以 GPS 作为基准让其他的传感器数据与其直接对齐，会产生比较大的误差，影响我们的预测精度。具体处理如下：

1. **数据选择**：将所有原始数据的路线图与方位角相对于时间的曲线绘出，舍去类似于图**??**所示的有害数据，这类数据名保存在 **./lists/bad_list.txt** 中。

2. **数据清洗**：原始数据中存在缺失值（-1），在插值前删去有缺失值的条目。

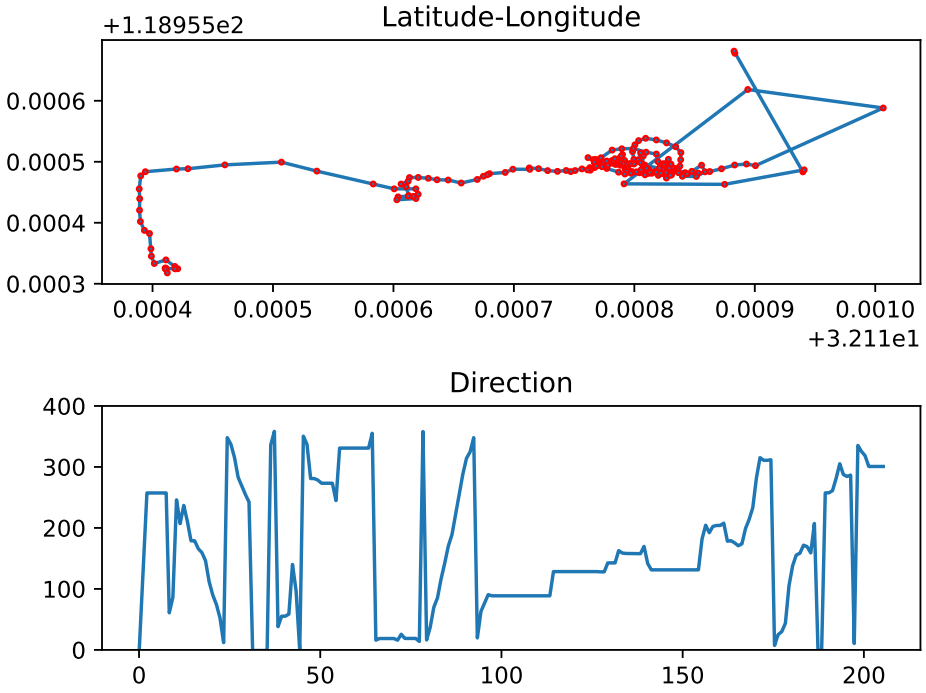


图 3: 有害数据示例图 (对应数据集 14)

3. **掐头去尾：**实验刚开始和结束前收集的数据误差较大，故舍去起始处与结束处的部分数据，此处封装为参数，可调节。

4. **角度转换：**角度范围是 0° 到 360°，故当实验人员保持 0° 方向行走时，由于数据测量波动，方位角可能在 >350 与 <10 间频繁波动，如图 **??** 所示，使方位角数据变得不连续，不利于后续处理。故在处理原始数据时，若当前数据方位角 <10 且前一条数据 >350，则将当前方位角设为 359；若当前方位角 >350 且前一条数据方位角 <10，则将当前方位角设为 0。

5. **数据同步：**观察原始数据，发现不同传感器输出的数据时间戳并不同步，为了达到能将多个设备的数据进行时间戳对齐的目标，我们使用 python 中 scipy 的一维插值方式对传感器和 GPS 测量结果进同步处理，使得数据属性和预测值的时间戳相同。最终数据统一为 50HZ.

### 2.2 RIDI Dataset

RIDI [1] 数据集包含加速度计、陀螺仪和磁力计的测量值与行人行进轨迹的定位，且根据不同设备状态分为手持、放在背包里、放在裤兜里，同时根据行人不同的速度划分为更加细致的亚类。该数据集记录了总长 150 分钟，传感器频率 200Hz 的数据。但由于

[1]由于该数据集较大，其压缩包大小约为 1.3G，因此我们并未上传该数据集，但我们提供了数据集的下载链接 https://box.nju.edu.cn/d/fa1e015b13654a56899f/

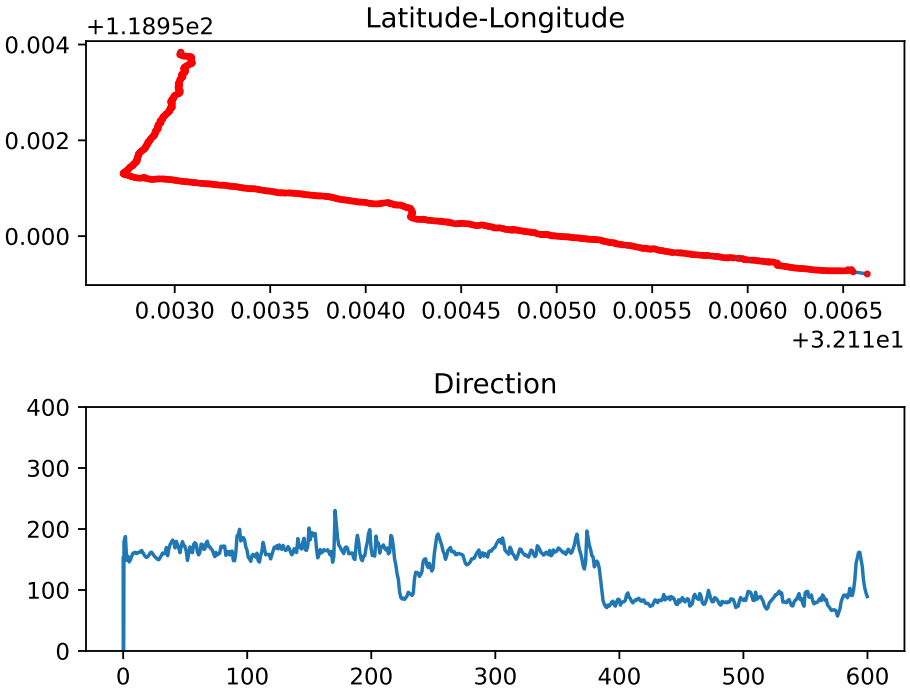


图 4: 正常数据示例图 (对应数据集 37)

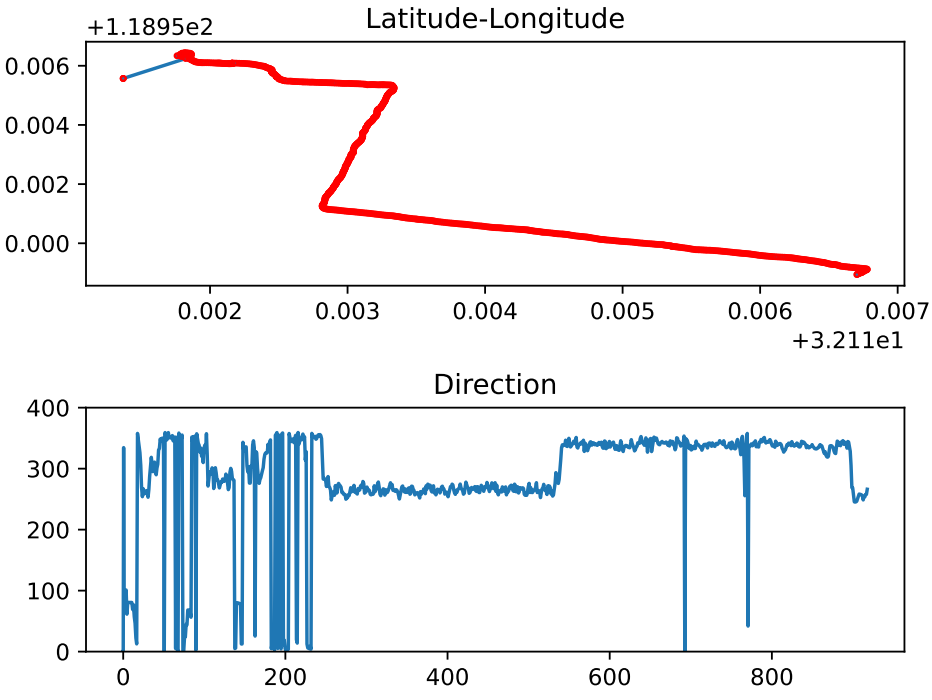


图 5: 方位角震荡示例图 (对应数据集 56)

该数据集包含的输出与我们需要的输出 (经纬度、方向角) 并不相同，因此该数据集只被我们用来进行迁移学习的训练。具体关于迁移学习的细节见第 ?? 节。

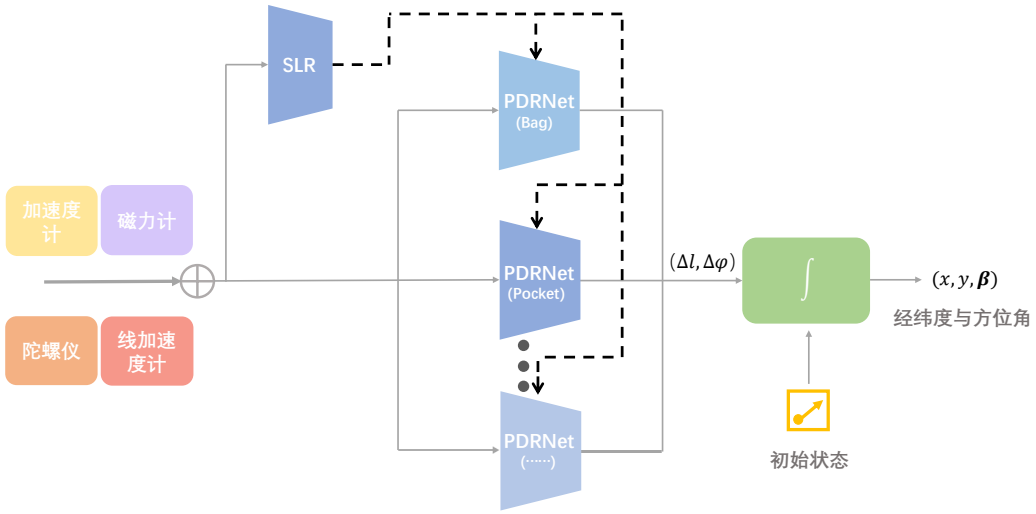


图 6: PDRNet 算法框架：将 PDR 问题分为两个步骤，即智能手机位置识别 SLR 和行人定位。SLR 网络预测智能手机位置，并相应地仅激活一个专用 PDRNet 网络（每个智能手机位置一个）；PDRNet 推断行人位置和方向的变化。

## 3 SOTA PDRNet

### 3.1 Basic Module

与 RoNIN 类似，我们使用了一维的卷积神经网络作为基础的位置预测模块，并采用了如今广泛使用的残差连接模式，以使得训练过程更加高效、训练模型的泛化性能更强。ResNet 有很多变体，由于我们有的训练数据数据量并不大，我们最终选择使用了轻量级的 ResNet-18。ResNet-18 由 8 个残差模块组成，其中每个残差模块包含卷积核大小为 3 的卷积层。

如第 ?? 节所说，我们收集到的数据中传感器的测量值 (即 input) 和 GPS (即 output) 的测量值的数值是未对齐的。其中传感器的测量值频率约为 50Hz，GPS 的测量频率约为 1Hz。因此我们将 50 组加速度计测量值 ($\mathbb{R}^3$)、陀螺仪测量值 ($\mathbb{R}^3$)、磁力计测量值 ($\mathbb{R}^3$) 拼接起来作为一个输入，对应一个由经纬度 ($\mathbb{R}^2$)、方位角 ($\mathbb{R}^1$) 组成的输出。更精确地，输入 $I \in \mathbb{R}^{450\times b}$，其中 $b$ 为 batch size；输出 $O \in \mathbb{R}^{3\times b}$。

ResNet 的训练目标函数 $\mathcal{L}$ 定义为经纬度增量值误差 $\Delta x$ 的 $L_2$ 范数与方向 $\beta$ 误差的 $L_2$ 范数的加权和，如公式 ?? 所示。其中，$\lambda$ 为经纬度与方向之间的权重。此外，我们使用 Adam 作为优化器以自动调节学习率。

$$\mathcal{L} = \frac{L_2(\Delta x, \Delta\hat{x}) + \lambda L_2(\beta, \hat{\beta})}{1+\lambda} \tag{1}$$

### 3.2 Transfer Learning

由于模型可能运行在不同的设备以及不同的情景下，因此我们仿照 ? 在基本的 PDRNet 上增加了一个手机场景识别模块，简称 SLR，以提升模型的泛化性能，达到模型迁移的效果。该 SLR 模块和基本的 PDRNet 的输入相同，输出为拿在手上、放在背包里、放在裤兜里的三个类别。该 SLR 分类模块采用一维卷积神经网络架构，并将卷积神经网络输出的 feature 经过 ReLU 激活函数后输入一个由一层全连接层和一个 softmax 层构成的分类器中。损失函数使用分类交叉熵（CCE），并使用 RMSProp 作为优化器。完整的算法框架见图示 **??**。

除了使用手机场景识别模块进行迁移学习之外，我们还尝试采用了预训练 + 微调的方法进行迁移。具体的，我们并不是使用各个场景各自的数据训练各自的基本 PDRNet 模块，而是将所有数据混合在一起预先训练出一个 PDRNet 模块，再分别在各自的场景的数据下进行 finetune。

## 4 Experiment & Analysis

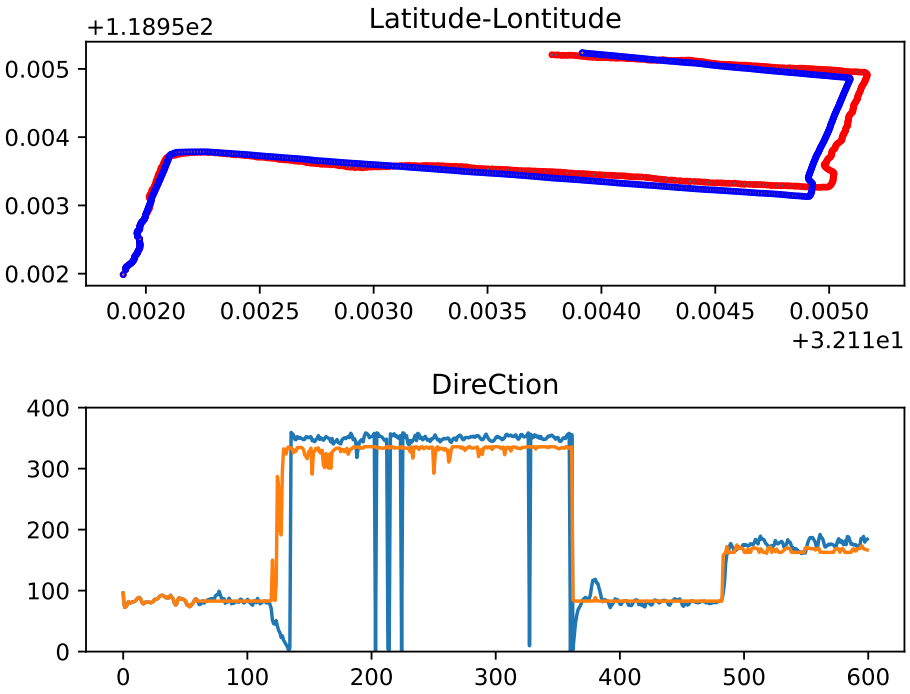


图 7: ENNRF 算法的预测效果图：其中上子图为经纬度的轨迹图，红色曲线为 Ground Truth 轨迹，蓝色曲线为预测轨迹；下子图为方位角的时间函数图，浅蓝色曲线为 Ground Truth 时间函数，橙色曲线为预测的方位角时间函数。

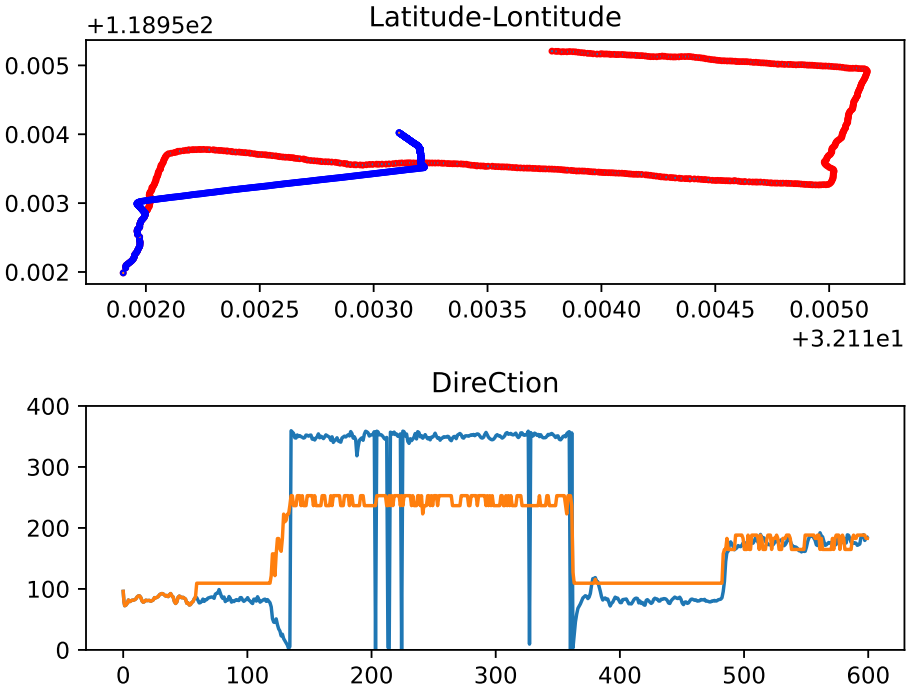


图 8: Adaboost 算法的预测效果图，图例同图 **??**

### 4.1 Baselines

除了我们开始尝试使用的 PDRNet 方法和最终选择使用的集成多层前馈神经网络和随机森林的方法外，我们还尝试了以下四种 Baseline 方法来作为性能上的比较，其中几种方法在提供的测试集 test0 上的预测误差见表 **??**。

**多元线性回归 (LR)：**使用简单的多元线性回归对于方位角及经纬度的增量分别进行拟合。输入为数据集中的特征，输出为需要预测的三个目标物理量。我们使用 Scikit-learn 中的 LinearRegression 进行拟合，拟合结果中，经纬度增量的均方误差达到 7261.01，方位角的均方误差 2378.23。由此可见线性模型并不适用于该问题，因此我们没有在此做过多的探索。

**支持向量机 (SVR)：**坐标距离误差 270.69 米，方位角误差 79.66°，效果与随机森林相比无优势，且训练速度过慢，舍弃。

**解析方法 (Analytic)：**传统的 PDR 算法利用物理公式直接计算方位角，故此处也进行尝试。发现方位角误差在 test case0 上可以降到 8.23，明显优于现有机器学习方法。遗憾的是非常不稳定，在不同数据集间误差波动较大，最终舍弃此种方法，转而尝试将其作为数据处理的方式，将原始特征进行转化，送入机器学习方法进行学习。

**自适应提升 (Adaboost)：**坐标距离误差 144.02 米，方位角误差 59.86°，效果与随机森林相比无优势，舍弃。

| 误差 ↓ | SVR | Adaboost | RF | FFN | ENNRF(Ours) |
|---|---|---|---|---|---|
| 经纬度 *(m)* | 270.69 | 144.02 | 46.54 | **8.89** | **8.89** |
| 方位角 *(°)* | 79.66 | 59.86 | **14.32** | 34.09 | **14.32** |

表 1: 各种方法在 test0 数据集上的误差结果比较 (粗体数字表示最小误差)

### 4.2 Final Method: ENNRF

我们最终决定使用集成前馈神经网络和随机森林的方法，其中 FFN 负责预测经纬度，随机森林负责预测方位角，ENNRF 最终的预测效果可视化见图 **??** 所示。作为对比，我们也绘制了 Adaboost 的预测效果图 **??**，以凸显我们方法的优越性。

#### 4.2.1 随机森林 (RF)

针对 PDR 中的方位角预测问题，随机森林模型主要以下有 5 种优点：

1. **抗过拟合能力强**：通过集成大量决策树，降低过拟合的风险性
2. **异常点不敏感**：只有在半数以上的基分类器出现差错时才会做出错误的预测，这意味着，即使数据集中出现了一个异常的数据点，整个算法也不会受到过多影响。因为它只会影响到部分决策树，很难对所有决策树产生影响。而受硬件条件的掣肘，我们的数据集中存在一定数量的异常点（比如-1 值），使用随机森林模型可以规避此风险。
3. **冗余特征不敏感**：能够处理 feature 较多或者 feature 强相关的数据，无需进行特征选择，因为单棵随机树的特征子集是随机选择的。本次任务有多个传感器，每个传感器给出一个或多个特征，显然难以做的相互独立，且由于缺乏先验知识，模型特征的选择可能非最优，使用随机森林模型可以规避此风险。
4. **泛化能力强**：对 generlization error 使用的是无偏估计，对迁移场景有利。
5. **训练速度快**：决策树间相互独立，可以并行化训练。

具体到训练时，我们首先对三个参数（决策树数量、最大深度、单棵树最大样本数）进行调节，发现三者分别为（400，5，6000）时效果较好，此时 test_case0 上方位角误差为 20°，多次训练波动不超过 0.5°。

接着尝试特征组合，最终发现使用 [gyrox, gyroy, magnetx, magnety, magnetz] 效果较好，test case0 上方位角误差为 14.3°，多次训练波动不超过 0.5°。分析原因，磁力计和陀螺仪已经可以反映方位角的变化，且只考虑平面情况，故不需要 Z 轴的陀螺仪数据。

在上文提及，使用物理公式计算方位角的方法在某些数据集上表现优异，但较不稳定，故想到可以将计算过程的中间变量作为输入随机森林的特征，利用加速度和磁力计数据根据公式计算 roll, pitch 和 yaw，与已有数据进行拼接。但是最终发现效果无明显

改善，且稳定性下降，舍弃此想法。

$$\text{roll} = \arctan\frac{a_x}{a_z} - \pi, \quad \text{pitch} = -\arctan\frac{a_y}{a_x \sin(\text{roll}) + a_z \cos(\text{roll})}$$

$$\text{yaw} = \arctan\left(\frac{m_x \sin(\text{roll}) \sin(pitch) + m_z \cos(\text{roll}) \sin(pitch) + m_y \cos(pitch) \cos(pitch)}{m_x \cos(\text{roll}) - m_z \sin(\text{roll})}\right) - \pi$$

#### 4.2.2 多层前馈神经网络 (FFN)

在该方法中，我们使用具有包含 32 个神经元的两个隐层的多层前馈神经网络模型处理该问题。我们令输入为数据集中传感器所提供的属性，输出为带预测的变量，我们企图通过训练网络使其能够根据瞬时的传感器速度预测方位的增量。该方法的尝试首先涉及特征和预测变量的选择，也就是解决“预测哪些变量?”,“用哪些变量来进行预测？”这两个问题。

对于以上两个问题我们通过排列组合不同的方案，对每个方案进行相同程度的训练对比结果可发现：使用**除压力计外的所有变量**预测**经纬度的增量**效果较好。

确定预测方案后，接下来就是进行进一步的训练，目的是达到在数据集上具有良好的泛化性能。然而该泛化性能分为两个部分，第一部分是在我们收集的数据集中划分出来的测试集 A，第二部分则是在作业中下发的 test0 测试集 (后记作测试集 B) 上的性能，在训练的时候我们分别计算两个测试集的误差来观察训练效果。

我们首先使用 Adam 优化器，将步长设置为 0.001 进行训练，发现训练到 200 个 epoch，测试集 A 上面的误差不断降低，然而测试集 B 的误差并没有具体的上升或下降趋势。我们推断可能体现在两个数据集的差异上。经过二维绘制我们发现，我们的测试集的行进曲线抖动明显且不平滑，然而作业中提供的数据集行进曲线简单平缓，我们推测训练的步长调的过大，同时训练轮数过多，导致神经网络学习到了训练集中高频的信息，然而这些信息不利于良好的泛化性能，于是我们将步长调小 10 倍，发现其在 200 轮的时候达到在测试集 B 上几乎收敛的效果。平均误差能够达到 20 以内。

## 5 Implementation

本次实验我们使用了 2 × 3090ti@Nvidia，以及内存为 32G 的 8 核 CPU。所有神经网络模型（ResNet、FFN、CNN）均使用 Pytorch 实现，随机森林、支持向量机、AdaBoost 等模型使用 scikit-learn 实现。

最终生成的十个预测结果保存在 **./outputs** 目录下; 预测方位角的 RF 模型保存在 **./models/rf** 目录下，源代码文件为 **./source/baseline/RF.py**，预测经纬度的 FFN 模型保存在 **./models/ffn** 目录下，源代码文件为 **./source/baseline/ffn_finetune.py**; 结果输出接口为 **./source/output_all.py**

## 6 Cooperation

- 赵思衡：负责项目的整体统筹，实验报告的撰写，部分代码的编写，包括：
  1. 项目准备阶段，进行大部分的文献调研，确定使用的模型方案；
  2. 项目初期，提供服务器与显卡，配置实验平台的环境；统一代码、文件、数据的编写、存储格式；
  3. 项目进行时，编写小部分实验代码，主要为 ResNet、PDRNet 的部分；撰写绝大部分的实验报告；
  4. 项目收尾，进行代码与数据的整合。
- 王科：初期进行了文献调研与数据采集，后期负责主要代码的编写并撰写对应部分的报告，包括：
  1. 参考调研得到的 RoNIN 模型架设项目文件框架；
  2. 确定数据处理方式，并以此为依据进行数据的选择、清洗、插值同步与数据加载接口的编写；撰写对应部分报告；
  3. 完成随机森林、SVR、AdaBoost 等模型的训练、评估与优化，最终选择使用随机森林预测方位角；进行解析公式计算与机器学习方法的对比并尝试将二者结合；撰写对应部分报告；
  4. 实现原始数据与预测结果可视化；负责最终模型整合与预测结果输出接口。
- 林滋芊：报告撰写，代码编写，数据整理，文献调研均有涉及，包括：
  1. 文献搜索与文献方法归纳；
  2. 数据收集整合，数据预处理工作；
  3. 多元线性回归模型与 FFN 模型的训练、评估与优化，最终选择使用 FFN 模型预测经纬度；
  4. 实验报告编写与图片绘制。