Setup

First wget ....amazonaws.com... (removed due to copyright issues)

In each data fold, there is a raw data subfolder and a syn data subfolder, which represent the raw data collection without synchronisation but with high precise timestep, and the synchronised data but without high precise timestep.

Here is the header of the sensor file and ground truth file.

vicon (vi*.csv)

Time Header translation.x translation.y translation.z rotation.x rotation.y rotation.z rotation.w

Sensors (imu*.csv)

Time attitude_roll(radians) attitude_pitch(radians) attitude_yaw(radians) rotation_rate_x(radians/s) rotation_rate_y(radians/s) rotation_rate_z(radians/s) gravity_x(G) gravity_y(G) gravity_z(G) user_acc_x(G) user_acc_y(G) user_acc_z(G) magnetic_field_x(microteslas) magnetic_field_y(microteslas) magnetic_field_z(microteslas)

Structure

In this folder

```
user@fedora ~/C/magnetic_localization (master)> ls data/Oxford\ Inertial\ 
imu1.csv*  imu4.csv*  imu7.csv*  vi3.csv*  vi6.csv*
imu2.csv*  imu5.csv*  vi1.csv*   vi4.csv*  vi7.csv*
imu3.csv*  imu6.csv*  vi2.csv*   vi5.csv*
user@fedora ~/C/magnetic_localization (master)> ls data/Oxford\ Inertial\ 
imu1.csv*  imu2.csv*  imu3.csv*  vi1.csv*  vi2.csv*  vi3.csv*
```

```
user@fedora ~/C/magnetic_localization (master)> ls data/Oxford\ Inertial\ (
data1/  data3/  data5/         Test.txt*
data2/  data4/  handheld.xlsx*  Train.txt*
user@fedora ~/C/magnetic_localization (master)> pwd
/home/user/Code/magnetic_localization
```

Also like each of them are the same length, so no need to sync the timesteps

```
user@fedora ~/C/magnetic_localization (master)> cat data/Oxford\ Inertial\
  23446   23446 3282548
user@fedora ~/C/magnetic_localization (master)> cat data/Oxford\ Inertial\
  23446   23446 1740520
```

Just ignore the Time and Header

Goal

Our goal is to predict the current `x`, `y`, `z` based on the previous all previous data(but not previous `x`, `y`, `z`)

Trying Vanilla LSTM

1. Suffers from distribution shift

```
X_train means:
mag_x: -0.46830051313300697
mag_y: -15.668869715403527
mag_z: -36.376663738555756
mag_total: 42.527113564066134

X_test means:
mag_x: -4.326749743812862
mag_y: -13.854210498185887
mag_z: -32.72580701915449
mag_total: 38.786503919304415
```

```
y_train means:
x: 0.12568006574318533
y: 0.023374721301369764
z: 1.176188457321701

y_test means:
x: 0.15600621631161352
y: 0.075468841401043
z: 1.1843440265075935
```

1. High variance

```
Epoch [0/49], Train MSE (denorm): 1.4946, Test MSE (denorm): 2.5259
Epoch [0/49], Train Loss: 0.3160
Epoch [1/49], Train MSE (denorm): 0.2647, Test MSE (denorm): 3.6559
Epoch [1/49], Train Loss: 0.2283
Epoch [2/49], Train MSE (denorm): 0.1589, Test MSE (denorm): 3.4956
```

VIT Approach(ToDo haven't implemented

Input Layer

- **Segmentation**: Divide your input sequence into fixed-size "patches" (e.g., 1-second windows).
- **Projection**: Project each patch to a higher-dimensional space using a linear transformation to create patch embeddings.

Positional Encoding

- **Encoding**: Add learnable or fixed positional embeddings to the patch embeddings to maintain temporal order.

Transformer Encoder

- **Structure**: Stack of Transformer encoder blocks (e.g., 6-12 layers).
- **Components**:

- Multi-Head Self-Attention
- Layer Normalization
- Feed-Forward Network
- Residual Connections

Global Average Pooling

- **Aggregation**: Aggregate information across all patches.

MLP Head

- **Prediction**: Final layers to predict translation (x, y, z).

What helped in sequence processing?

By making sure sliding windows doesn't overlap, we saw a massive gain.

Easy LSTM

So basically this is an intuitive file for LSTM, with minimal configurations and it can be trained 5 minutes on a CPU

We basically did a really simple thing, like feed everything into LSTM model (split data on the file level)

Handheld Training

As we can see it quickly reaches some benchmark(though not bad)

```
Total training samples: 6002
Total validation samples: 981
Total samples: 6983
Input shape: torch.Size([100, 15])
Target shape: torch.Size([3])
Number of training batches: 188
Number of validation batches: 31
Performing mean baseline evaluation...
Baseline Train Loss: 1.5063, Baseline Val Loss: 1.6076
Epoch [1/50], Train Loss: 1.0333, Val Loss: 0.7039
Epoch [2/50], Train Loss: 0.6933, Val Loss: 0.7423
Epoch [3/50], Train Loss: 0.5872, Val Loss: 0.6355
Epoch [4/50], Train Loss: 0.5469, Val Loss: 0.5512
Epoch [5/50], Train Loss: 0.6917, Val Loss: 0.6491
Epoch [6/50], Train Loss: 0.5254, Val Loss: 0.5288
Epoch [7/50], Train Loss: 0.4866, Val Loss: 0.4836
Epoch [8/50], Train Loss: 0.4751, Val Loss: 0.5284
Epoch [9/50], Train Loss: 0.4397, Val Loss: 0.5155
Epoch [10/50], Train Loss: 0.5113, Val Loss: 0.5460
Epoch [11/50], Train Loss: 0.4426, Val Loss: 0.4561
Epoch [12/50], Train Loss: 0.4397, Val Loss: 0.5362
Epoch [13/50], Train Loss: 0.4454, Val Loss: 0.6536
Epoch [14/50], Train Loss: 0.4092, Val Loss: 0.4735
Epoch [15/50], Train Loss: 0.3930, Val Loss: 0.4855
Epoch [16/50], Train Loss: 0.3816, Val Loss: 0.3979
Epoch [17/50], Train Loss: 0.3616, Val Loss: 0.4172
Epoch [18/50], Train Loss: 0.3498, Val Loss: 0.4247
Epoch [19/50], Train Loss: 0.3768, Val Loss: 0.3786
Epoch [20/50], Train Loss: 0.3466, Val Loss: 0.3950
Epoch [21/50], Train Loss: 0.3492, Val Loss: 0.4395
Epoch [22/50], Train Loss: 0.3466, Val Loss: 0.4699
Epoch [23/50], Train Loss: 0.3274, Val Loss: 0.4523
Epoch [24/50], Train Loss: 0.3463, Val Loss: 0.3629
Epoch [25/50], Train Loss: 0.3294, Val Loss: 0.4008
Epoch [26/50], Train Loss: 0.3051, Val Loss: 0.3653
Epoch [27/50], Train Loss: 0.3010, Val Loss: 0.3548
```

Variance Too High

If we use the sliding window approach we can easily like make sure they don't overlap to make variance much smaller.