Setup

First wget https://s3.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.

In Google Colab

Open in Colab

Here I also changed the Oxford Datasets, so I seperated the Oxford into 2 folders for train and test instead of putting the information in each folder.

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

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\ (
data1/  data3/  data5/         Test.txt*
data2/  data4/  handheld.xlsx*  Train.txt*
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\ (
imu1.csv*  imu3.csv*  imu5.csv*  vi2.csv*  vi4.csv*
imu2.csv*  imu4.csv*  vi1.csv*   vi3.csv*  vi5.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 fucking ignore the Time and Header

Some more Information

```
user@fedora ~/C/m/d/O/h/d/syn (master)> pwd
```

```
/home/user/Code/magnetic_localization/data/Oxford Inertial Odometry Datase
user@fedora ~/C/m/d/O/h/d/syn (master)> ls
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/m/d/O/h/d/syn (master)> cat imu1.csv |head -n 5
1.50E+11,0.003649,0.44925,-0.21255,0.036483,-0.029496,0.020632,0.003286,-0
1.50E+11,0.00305,0.44954,-0.21219,0.067307,-0.038284,0.029241,0.002747,-0.
1.50E+11,0.002363,0.45003,-0.21184,0.076935,-0.039423,0.021788,0.002128,-0
1.50E+11,0.001778,0.45053,-0.21167,0.066339,-0.039321,0.006826,0.001601,-0
1.50E+11,0.001393,0.45086,-0.21171,0.037685,-0.030543,-0.010317,0.001253,-
user@fedora ~/C/m/d/O/h/d/syn (master)> cat vi1.csv |head -n 5
1.50E+11,12978,-1.2991,1.7212,1.1931,-0.21409,-0.012459,-0.097183,0.97189
1.50E+11,12979,-1.2993,1.7213,1.1932,-0.21473,-0.01237,-0.097239,0.97174
1.50E+11,12980,-1.2993,1.7213,1.1932,-0.21509,-0.012288,-0.097244,0.97166
1.50E+11,12981,-1.2994,1.7214,1.1933,-0.21526,-0.012291,-0.09738,0.97161
1.50E+11,12982,-1.2995,1.7213,1.1933,-0.21541,-0.012199,-0.097503,0.97157
```

Note: Use os list dir instead of hardcoding data folders

YOU MUST SPLIT DATA AT THE FILE LEVEL TO PREVENT LEAKAGE

TENSORBOARD Logs

Tensorboard logs shouldn't be put in Git, so I put them here

`https://jimchen4214-public.s3.amazonaws.com/other/mag_tensorboard_logs/logs.zip` `https://jimchen4214-public.s3.amazonaws.com/other/mag_tensorboard_logs/lstm_logs.zip`

Command to upload

```
zip -r logs.zip logs/
aws s3 cp logs.zip s3://jimchen4214-public/other/mag_tensorboard_logs/logs
```

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 without splitting windows

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
```

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)

Improvements

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.

Sequence Too Short

If the sequence is too short then it performs poorly, improving the sequence length to 200 or 300 drastically improves the performance.

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 [10/50], Train Loss: 0.5113, Val Loss: 0.5460
Epoch [15/50], Train Loss: 0.3930, Val Loss: 0.4855
Epoch [20/50], Train Loss: 0.3466, Val Loss: 0.3950
Epoch [25/50], Train Loss: 0.3294, Val Loss: 0.4008
Epoch [27/50], Train Loss: 0.3010, Val Loss: 0.3548
```

Tried it on Trolley

```
(mlenv) user@fedora ~/C/m/src (master)> python easy_lstm.py
Total training samples: 3880
Total validation samples: 376
Total samples: 4256
Input shape: torch.Size([100, 15])
Target shape: torch.Size([3])
Number of training batches: 122
Number of validation batches: 12
Performing mean baseline evaluation...
Baseline Train Loss: 1.6448, Baseline Val Loss: 1.4999
Epoch [5/50], Train Loss: 0.5197, Val Loss: 0.4581
Epoch [10/50], Train Loss: 0.5856, Val Loss: 0.4645
Epoch [15/50], Train Loss: 0.4282, Val Loss: 0.4330
Epoch [20/50], Train Loss: 0.4565, Val Loss: 0.4461
Epoch [25/50], Train Loss: 0.3967, Val Loss: 0.3775
Epoch [29/50], Train Loss: 0.3408, Val Loss: 0.3440
```

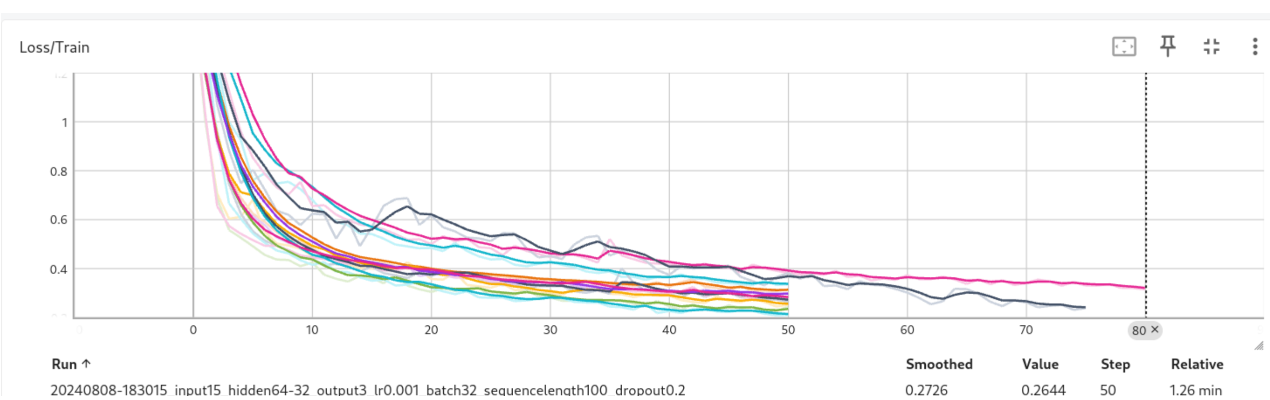
Finetuning

We need to finetune these

Basically I run these to test

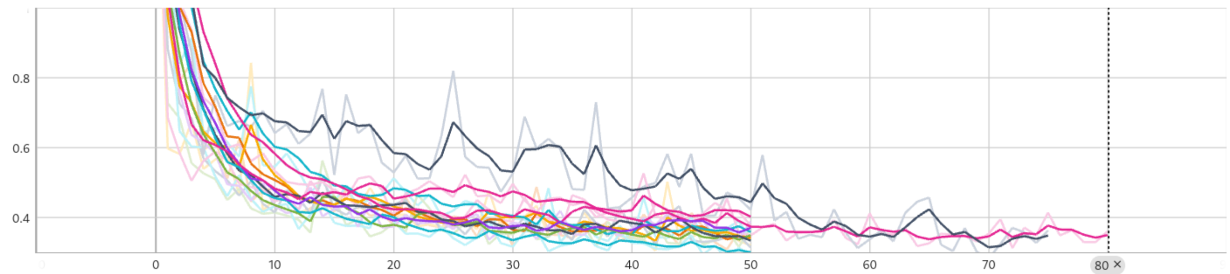
```
# Basic configuration:
python lstm_train.py --sequence_length 100 --hidden_sizes 64 32 --num_epoc
# Longer sequence length
python lstm_train.py --sequence_length 200 --hidden_sizes 64 32 --num_epoc
# Shorter sequence length
python lstm_train.py --sequence_length 50 --hidden_sizes 64 32 --num_epoch
# Single layer LSTM
python lstm_train.py --sequence_length 100 --hidden_sizes 128 --num_epochs
# Three-layer LSTM
python lstm_train.py --sequence_length 100 --hidden_sizes 64 32 16 --num_e
# Larger hidden sizes
python lstm_train.py --sequence_length 100 --hidden_sizes 128 64 --num_epo
# Smaller hidden sizes
python lstm_train.py --sequence_length 100 --hidden_sizes 32 16 --num_epoc
# Complex configuration
python lstm_train.py --sequence_length 300 --hidden_sizes 128 64 32 16 --n
###### With higher dropout #####
# Longer sequence length
python lstm_train.py --sequence_length 200 --hidden_sizes 64 32 --num_epoc

# Single layer LSTM
python lstm_train.py --sequence_length 200 --hidden_sizes 64 --num_epochs
```

Tensorboard Results



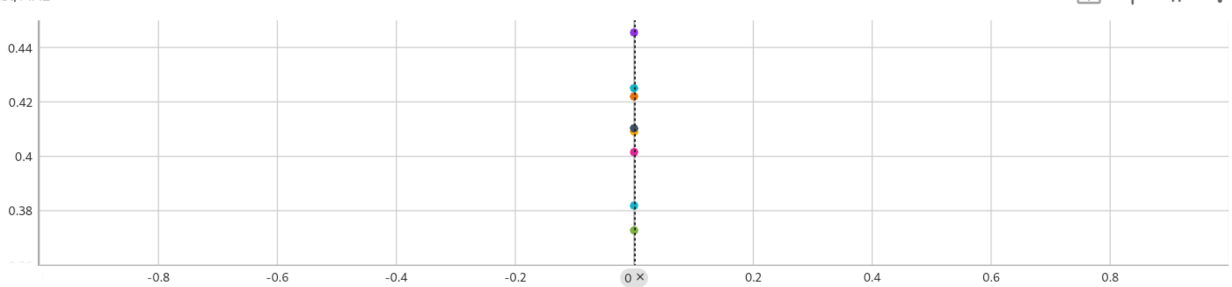
20240808-183257_input15_hidden64-32_output3_lr0.001_batch32_sequencelength200_dropout0.2	0.2141	0.2097	50	1.066 min
20240808-183609_input15_hidden64-32_output3_lr0.001_batch32__sequencelength50_dropout0.2	0.2837	0.2777	50	1.1 min
20240808-183823_input15_hidden128_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.2559	0.2513	50	1.476 min
20240808-184100_input15_hidden64-32-16_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.2973	0.2988	50	1.613 min
20240808-184442_input15_hidden128-64_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.2348	0.2427	50	2.473 min
20240808-184909_input15_hidden32-16_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.313	0.3149	50	44.19 sec
20240808-185123_input15_hidden128-64-32-16_output3_lr0.001_batch32_sequencelength300_dropout0.2	0.2416	0.2383	75	5.423 min
20240808-190423_input15_hidden64-32_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.3377	0.3355	50	1.637 min
20240808-212038_input15_hidden32_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.3211	0.3143	80	46.6 sec

Loss/Validation



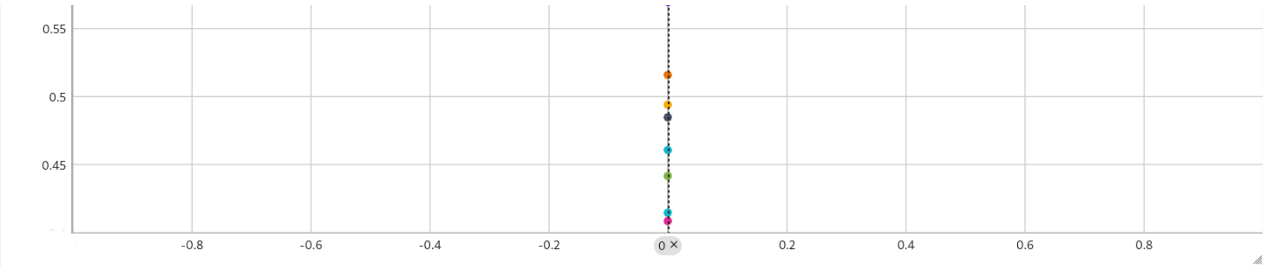
Run ↑	Smoothed	Value	Step	Relative
20240808-183015_input15_hidden64-32_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.3336	0.3136	50	1.26 min
20240808-183257_input15_hidden64-32_output3_lr0.001_batch32_sequencelength200_dropout0.2	0.3003	0.2912	50	1.066 min
20240808-183609_input15_hidden64-32_output3_lr0.001_batch32__sequencelength50_dropout0.2	0.4021	0.377	50	1.1 min
20240808-183823_input15_hidden128_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.3367	0.3204	50	1.476 min
20240808-184100_input15_hidden64-32-16_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.3697	0.3889	50	1.613 min
20240808-184442_input15_hidden128-64_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.3492	0.3685	50	2.473 min
20240808-184909_input15_hidden32-16_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.3433	0.3397	50	44.19 sec
20240808-185123_input15_hidden128-64-32-16_output3_lr0.001_batch32_sequencelength300_dropout0.2	0.3482	0.3627	75	5.423 min
20240808-190423_input15_hidden64-32_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.3581	0.3331	50	1.637 min
20240808-212038_input15_hidden32_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.3495	0.3606	80	46.6 sec

Test/MAE



Run ↑	Smoothed	Value	Step	Relative
20240808-183015_input15_hidden64-32_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.4103	0.4103	0	0
20240808-183257_input15_hidden64-32_output3_lr0.001_batch32_sequencelength200_dropout0.2	0.3818	0.3818	0	0
20240808-183609_input15_hidden64-32_output3_lr0.001_batch32__sequencelength50_dropout0.2	0.5009	0.5009	0	0
20240808-183823_input15_hidden128_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.4091	0.4091	0	0
20240808-184100_input15_hidden64-32-16_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.4455	0.4455	0	0
20240808-184442_input15_hidden128-64_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.3727	0.3727	0	0
20240808-184909_input15_hidden32-16_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.422	0.422	0	0
20240808-185123_input15_hidden128-64-32-16_output3_lr0.001_batch32_sequencelength300_dropout0.2	0.3445	0.3445	0	0
20240808-190423_input15_hidden64-32_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.4251	0.4251	0	0
20240808-212038_input15_hidden32_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.4015	0.4015	0	0

Test/MSE



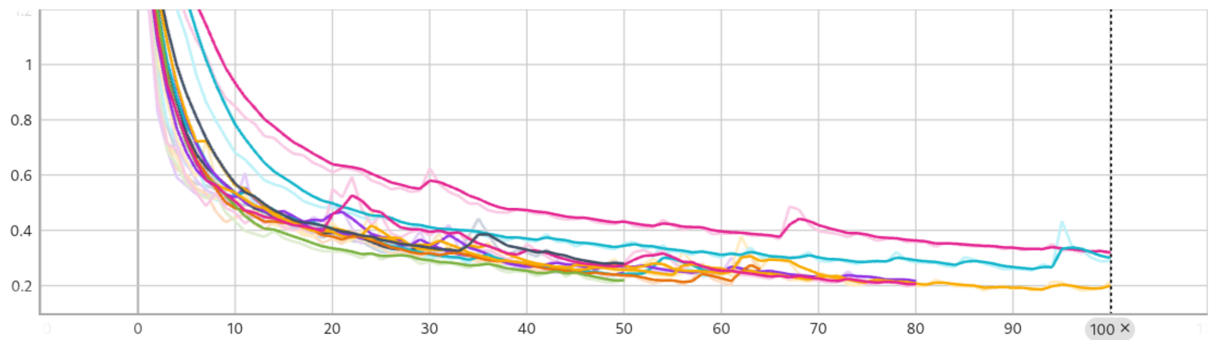
Run ↑	Smoothed	Value	Step	Relative
20240808-183015_input15_hidden64-32_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.4848	0.4848	0	0
20240808-183257_input15_hidden64-32_output3_lr0.001_batch32_sequencelength200_dropout0.2	0.4147	0.4147	0	0
20240808-183609_input15_hidden64-32_output3_lr0.001_batch32__sequencelength50_dropout0.2	0.7172	0.7172	0	0
20240808-183823_input15_hidden128_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.4941	0.4941	0	0
20240808-184100_input15_hidden64-32-16_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.5697	0.5697	0	0
20240808-184442_input15_hidden128-64_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.4417	0.4417	0	0
20240808-184909_input15_hidden32-16_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.516	0.516	0	0
20240808-185123_input15_hidden128-64-32-16_output3_lr0.001_batch32_sequencelength300_dropout0.2	0.3322	0.3322	0	0
20240808-190423_input15_hidden64-32_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.4607	0.4607	0	0
20240808-212038_input15_hidden32_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.4085	0.4085	0	0

Some More Training and Finetuning

```
python lstm_train.py --sequence_length 200 --hidden_sizes 32 --num_epochs
python lstm_train.py --sequence_length 200 --hidden_sizes 32 --num_epochs
# Large Batch
python lstm_train.py --sequence_length 200 --hidden_sizes 32 --num_epochs
python lstm_train.py --sequence_length 200 --hidden_sizes 32 --num_epochs
# Large Hidden Size
python lstm_train.py --sequence_length 200 --hidden_sizes 64 --num_epochs

# Run it twice
python lstm_train.py --sequence_length 200 --hidden_sizes 64 --num_epochs
python lstm_train.py --sequence_length 200 --hidden_sizes 64 --num_epochs

python lstm_train.py --sequence_length 200 --hidden_sizes 256 --num_epochs

# More Runs
python lstm_train.py --sequence_length 200 --hidden_sizes 64 --num_epochs
python lstm_train.py --sequence_length 200 --hidden_sizes 96 --num_epochs
python lstm_train.py --sequence_length 200 --hidden_sizes 128 64 --num_epo
```

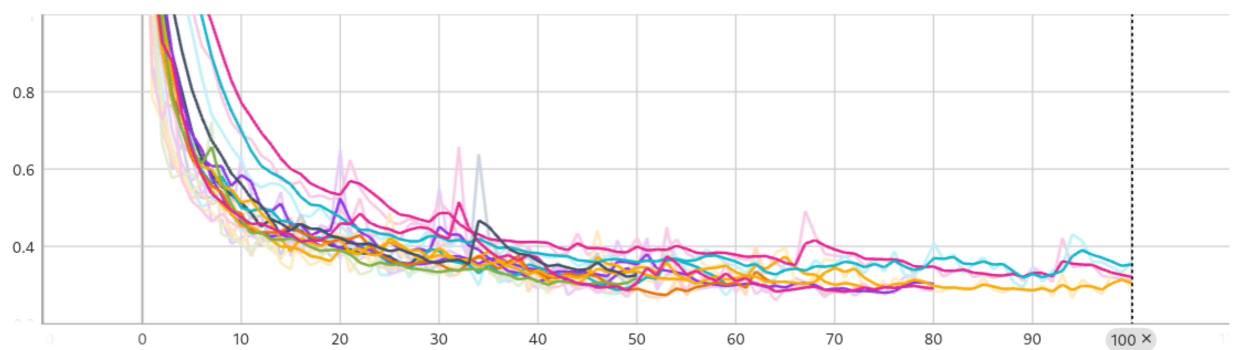
Tensorboard Results

Loss/Train



Run ↑	Smoothed	Value	Step	Relative
20240808-212646_input15_hidden32_output3_lr0.001_batch32_dropout0.2_sequencelength200	0.2797	0.2764	50	27.4 sec
20240808-213347_input15_hidden32_output3_lr0.001_batch128_dropout0.2_sequencelength200	0.3004	0.2897	100	54.58 sec
20240808-213448_input15_hidden32_output3_lr0.001_batch128_dropout0.4_sequencelength200	0.3198	0.3124	100	1.368 min
20240808-214339_input15_hidden64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.1994	0.2136	100	1.685 min
20240808-215332_input15_hidden256_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2464	0.2572	50	4.384 min
20240808-220141_input15_hidden64_output3_lr0.001_batch32_dropout0.2_sequencelength200	0.219	0.2195	50	54.03 sec
20240808-221113_input15_hidden96_output3_lr0.001_batch32_dropout0.35_sequencelength200	0.2576	0.2323	64	1.95 min
20240808-221357_input15_hidden64_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.3369	0.3487	29	29.08 sec
20240808-221511_input15_hidden64_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.2424	0.2334	60	54.54 sec
20240808-221726_input15_hidden96_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2047	0.2034	80	2.472 min
20240808-222130_input15_hidden128-64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2107	0.2067	75	3.885 min
20240808-222532_input15_hidden64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2159	0.2076	80	1.6 min

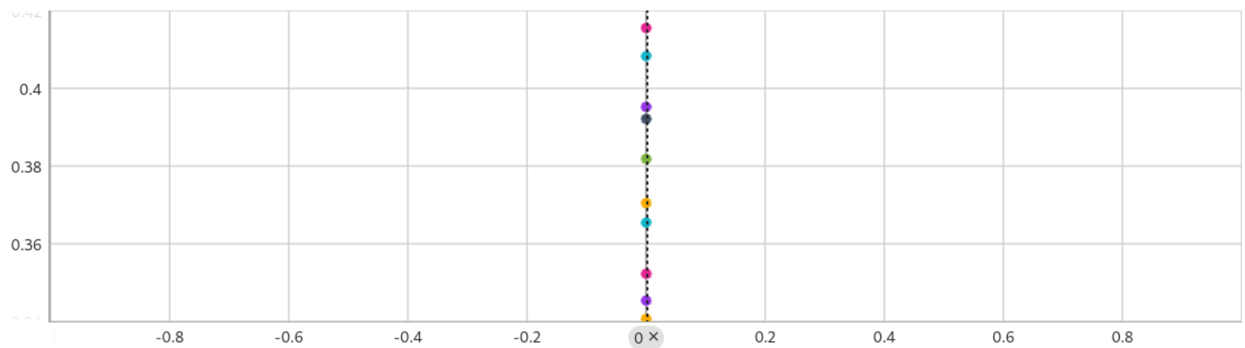
Loss/Validation



Run ↑	Smoothed	Value	Step	Relative
20240808-212646_input15_hidden32_output3_lr0.001_batch32_dropout0.2_sequencelength200	0.3254	0.3323	50	27.4 sec
20240808-213347_input15_hidden32_output3_lr0.001_batch128_dropout0.2_sequencelength200	0.353	0.3591	100	54.58 sec
20240808-213448_input15_hidden32_output3_lr0.001_batch128_dropout0.4_sequencelength200	0.3205	0.3123	100	1.368 min
20240808-214339_input15_hidden64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3047	0.2914	100	1.685 min
20240808-215332_input15_hidden256_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3279	0.3049	50	4.384 min
20240808-220141_input15_hidden64_output3_lr0.001_batch32_dropout0.2_sequencelength200	0.318	0.3372	50	54.03 sec
20240808-221113_input15_hidden96_output3_lr0.001_batch32_dropout0.35_sequencelength200	0.3024	0.2828	64	1.95 min
20240808-221357_input15_hidden64_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.3734	0.4027	28	28.13 sec

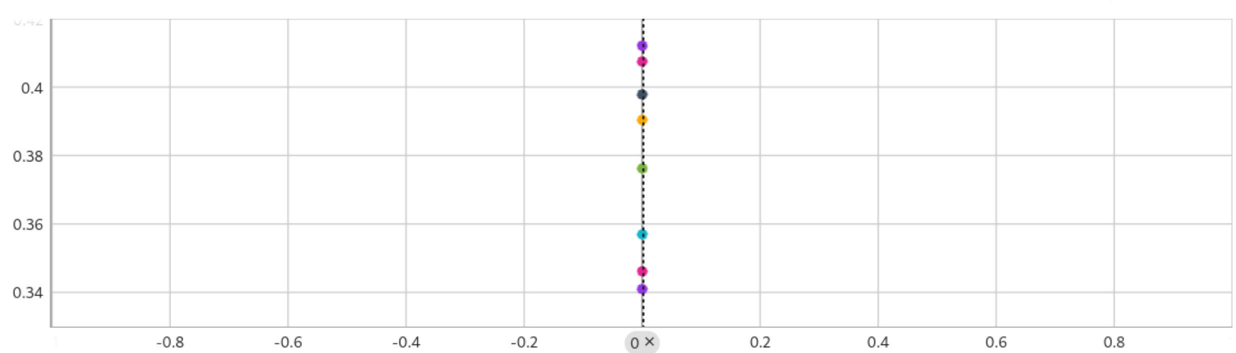
20240808-221511_input15_hidden64_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.303	0.3053	60	54.54 sec
20240808-221726_input15_hidden96_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2903	0.2857	80	2.472 min
20240808-222130_input15_hidden128-64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2986	0.3128	75	3.885 min
20240808-222532_input15_hidden64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.302	0.2947	80	1.6 min

Test/MAE



Run ↑	Smoothed	Value	Step	Relative
20240808-212646_input15_hidden32_output3_lr0.001_batch32_dropout0.2_sequencelength200	0.3921	0.3921	0	0
20240808-213347_input15_hidden32_output3_lr0.001_batch128_dropout0.2_sequencelength200	0.4083	0.4083	0	0
20240808-213448_input15_hidden32_output3_lr0.001_batch128_dropout0.4_sequencelength200	0.4155	0.4155	0	0
20240808-214339_input15_hidden64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3406	0.3406	0	0
20240808-215332_input15_hidden256_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3952	0.3952	0	0
20240808-220141_input15_hidden64_output3_lr0.001_batch32_dropout0.2_sequencelength200	0.3818	0.3818	0	0
20240808-221511_input15_hidden64_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.3654	0.3654	0	0
20240808-221726_input15_hidden96_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3522	0.3522	0	0
20240808-222130_input15_hidden128-64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3704	0.3704	0	0
20240808-222532_input15_hidden64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3453	0.3453	0	0

Test/MSE



Run ↑	Smoothed	Value	Step	Relative
20240808-212646_input15_hidden32_output3_lr0.001_batch32_dropout0.2_sequencelength200	0.3979	0.3979	0	0
20240808-213347_input15_hidden32_output3_lr0.001_batch128_dropout0.2_sequencelength200	0.4335	0.4335	0	0
20240808-213448_input15_hidden32_output3_lr0.001_batch128_dropout0.4_sequencelength200	0.4075	0.4075	0	0
20240808-214339_input15_hidden64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3285	0.3285	0	0
20240808-215332_input15_hidden256_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.4121	0.4121	0	0
20240808-220141_input15_hidden64_output3_lr0.001_batch32_dropout0.2_sequencelength200	0.3762	0.3762	0	0
20240808-221511_input15_hidden64_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.357	0.357	0	0
20240808-221726_input15_hidden96_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3462	0.3462	0	0

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
20240808-222130_input15_hidden128-64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3904	0.3904	0	0
20240808-222532_input15_hidden64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.341	0.341	0	0

Extreme Long Seq Length

I also tried to increase the sequence length to 400, which resulted in

```
Overall Mean Squared Error: 0.2241
Overall Mean Absolute Error: 0.2934
```

So like I cannot get it much below 0.3 with this vanilla LSTM approach

Easy Transformer

Transformer is more complicated to implement than RNN, and thus very prune to mistakes. I am workng hard to make sure no misconfigs happen(though it is very likely).

Run Some Configuarations

So basically run different configs to get an idea what happens

```
# Baseline configuration
python transformer_train.py --d_model 32 --nhead 2 --num_layers 1 --dim_fe

# Deeper model
python transformer_train.py --d_model 64 --nhead 4 --num_layers 2 --dim_fe
python transformer_train.py --d_model 64 --nhead 4 --num_layers 6 --dim_fe

# Wider model
python transformer_train.py --d_model 128 --nhead 8 --num_layers 2 --dim_f

# Mean pooling
python transformer_train.py --d_model 64 --nhead 4 --num_layers 2 --dim_fe

# Return all positions
```

```
python transformer_train.py --d_model 64 --nhead 4 --num_layers 2 --dim_fe

# Larger batch size
python transformer_train.py --d_model 64 --nhead 4 --num_layers 2 --dim_fe

# More training epochs
python transformer_train.py --d_model 64 --nhead 4 --num_layers 2 --dim_fe

# Higher dropout
python transformer_train.py --d_model 64 --nhead 4 --num_layers 3 --dim_fe
```