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

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
```

A list of things I am not doing

1. I am not implementing variants of LSTM or RNN because I don't think it's that meaningful, you can do it if you want
2. I don't think pure TCN is going to work. Maybe utilizing parallel CNNs for input layer might be a good idea.
3. I am not changing the `Bx`, `By` `Bz` into other forms because I think they yield similar results with all these models(the models automatically fits it), you can do it if you want
4. I am using the layernorm in the start of the model instead of scaling it manually

Tensorboard data on S3

```
https://jimchen4214-public.s3.us-east-1.amazonaws.com/other/transformer_lo
https://jimchen4214-public.s3.us-east-1.amazonaws.com/other/lstm_logs.zip
https://jimchen4214-public.s3.us-east-1.amazonaws.com/other/resnet_logs.zi
https://jimchen4214-public.s3.us-east-1.amazonaws.com/other/all_logs.zip
```

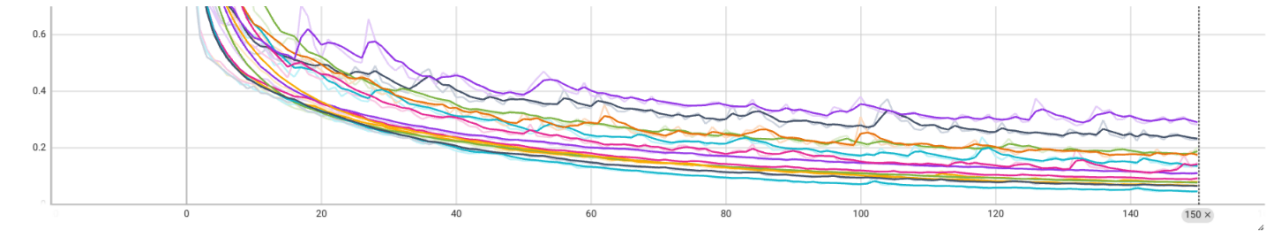
Final Training

I added up the Handheld, Trolley, Running, Walking, Handbag, Pocket, and used all my models to train it

See `run.sh`

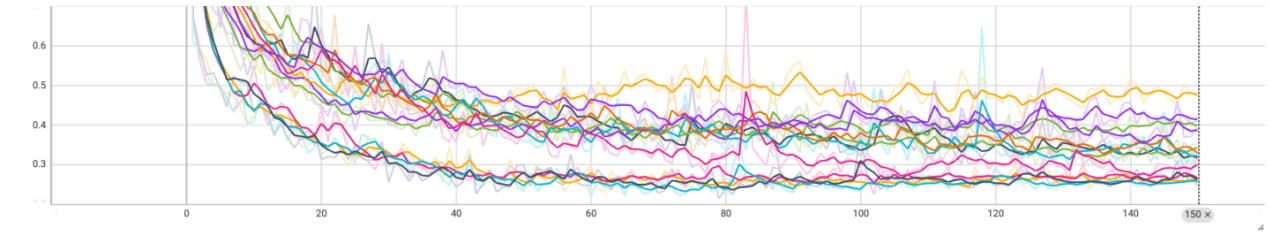
Results

Loss/Train



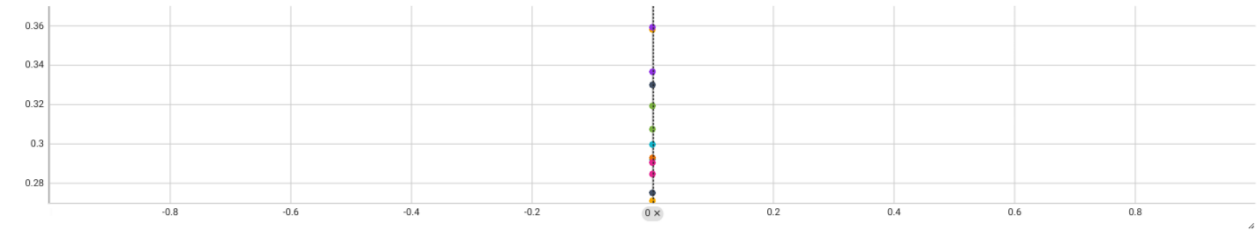
Run	Smoothed	Value	Step	Relative
resnet_20240814-034312_input12_channels128-256-512_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.0464	0.0465	150	7.071 min
resnet_20240814-033242_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.0654	0.064	150	6.245 min
transformer_20240814-055924_d_model128_nhead8_layers2_ff1024_output3_batch32_dropout0.3_sequencelength200_poolinglast_returnallFalse	0.0667	0.0653	150	8.13 min
transformer_20240814-061805_d_model64_nhead4_layers2_ff256_output3_batch32_dropout0.2_sequencelength200_poolinglast_returnallTrue	0.0776	0.0768	150	5.466 min
resnet_20240814-040513_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.0797	0.0798	150	6.324 min
resnet_20240814-035439_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.092	0.0959	150	6.291 min
transformer_20240814-061004_d_model64_nhead4_layers2_ff256_output3_batch32_dropout0.2_sequencelength200_poolinglast_returnallFalse	0.1107	0.1123	150	5.489 min
lstm_20240814-045702_input12_hidden256-128_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.1368	0.1348	150	11.17 min
lstm_20240814-051637_input12_hidden512-256_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.1424	0.1476	150	26.61 min
lstm_20240814-043023_input12_hidden256-128_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.1762	0.1705	150	11.27 min
lstm_20240814-042126_input12_hidden128-64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.1884	0.2015	150	5.232 min
lstm_20240814-045009_input12_hidden128_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2329	0.2284	150	3.821 min
lstm_20240814-041551_input12_hidden64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.291	0.2822	150	3.534 min

Loss/Validation



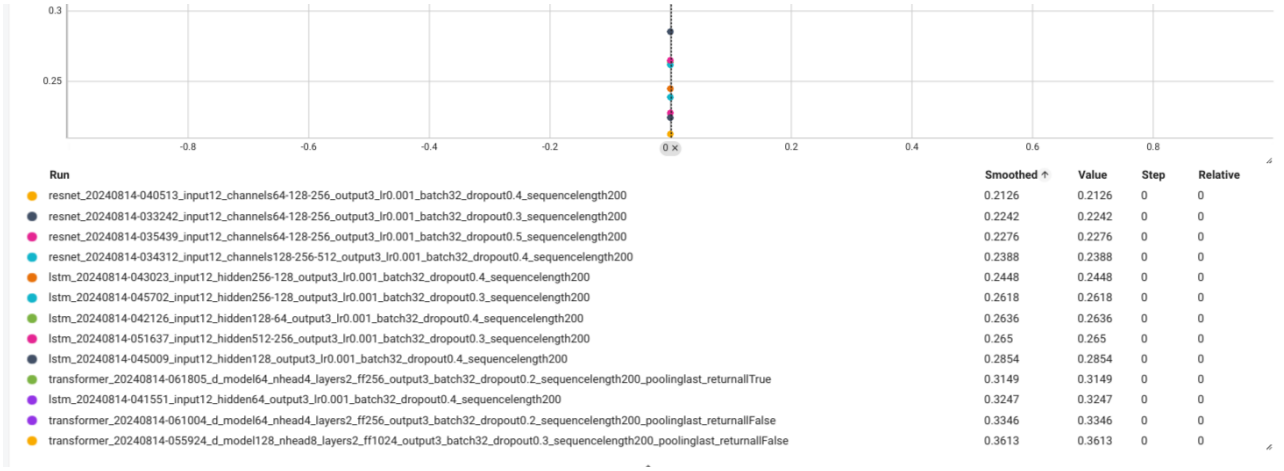
Run	Smoothed	Value	Step	Relative
resnet_20240814-034312_input12_channels128-256-512_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2595	0.2611	150	7.071 min
resnet_20240814-033242_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.2624	0.2471	150	6.245 min
resnet_20240814-040513_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2628	0.2656	150	6.324 min
resnet_20240814-035439_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.2671	0.2551	150	6.291 min
lstm_20240814-051637_input12_hidden512-256_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.3005	0.3169	150	26.61 min
lstm_20240814-045009_input12_hidden128_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.318	0.3115	150	3.821 min
lstm_20240814-045702_input12_hidden256-128_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.3227	0.3247	150	11.17 min
lstm_20240814-043023_input12_hidden256-128_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3295	0.3094	150	11.27 min
lstm_20240814-042126_input12_hidden128-64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3432	0.3523	150	5.232 min
lstm_20240814-041551_input12_hidden64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3876	0.392	150	3.534 min
transformer_20240814-061805_d_model64_nhead4_layers2_ff256_output3_batch32_dropout0.2_sequencelength200_poolinglast_returnallTrue	0.3979	0.3878	150	5.466 min
transformer_20240814-061004_d_model64_nhead4_layers2_ff256_output3_batch32_dropout0.2_sequencelength200_poolinglast_returnallFalse	0.4146	0.4119	150	5.489 min
transformer_20240814-055924_d_model128_nhead8_layers2_ff1024_output3_batch32_dropout0.3_sequencelength200_poolinglast_returnallFalse	0.4773	0.472	150	8.13 min

Test/MAE

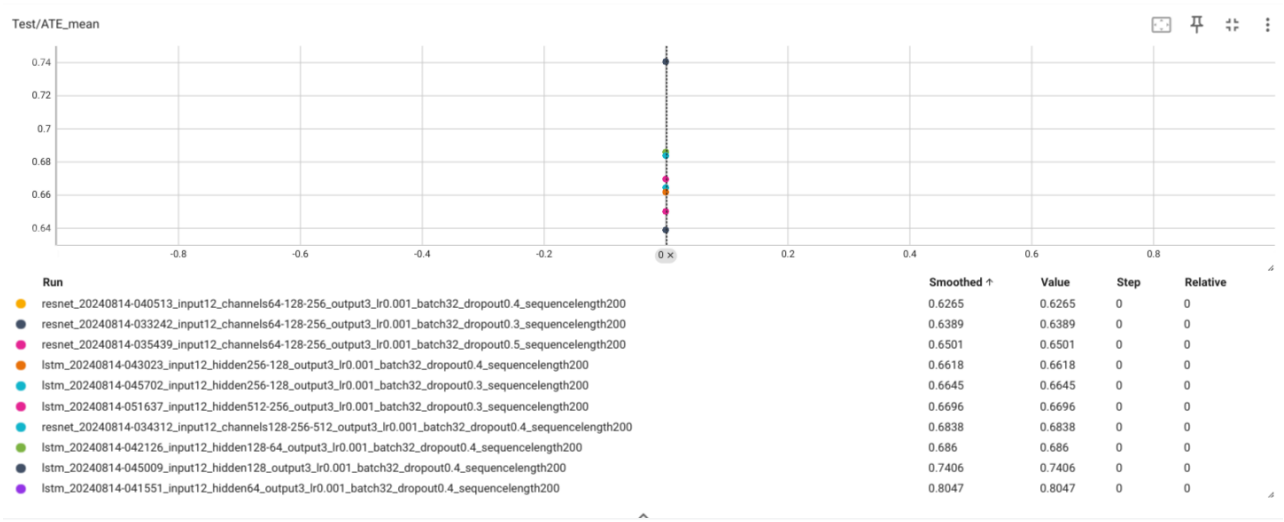


Run	Smoothed	Value	Step	Relative
lstm_20240814-041551_input12_hidden64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3593	0.3593	0	0
lstm_20240814-042126_input12_hidden128-64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3075	0.3075	0	0
lstm_20240814-043023_input12_hidden256-128_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2929	0.2929	0	0
lstm_20240814-045009_input12_hidden128_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.33	0.33	0	0
lstm_20240814-045702_input12_hidden256-128_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.2922	0.2922	0	0
lstm_20240814-051637_input12_hidden512-256_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.2905	0.2905	0	0
resnet_20240814-033242_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.2751	0.2751	0	0
resnet_20240814-034312_input12_channels128-256-512_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2997	0.2997	0	0
resnet_20240814-035439_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.2847	0.2847	0	0
resnet_20240814-040513_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2711	0.2711	0	0
transformer_20240814-055924_d_model128_nhead8_layers2_ff1024_output3_batch32_dropout0.3_sequencelength200_poolinglast_returnallFalse	0.3582	0.3582	0	0
transformer_20240814-061004_d_model64_nhead4_layers2_ff256_output3_batch32_dropout0.2_sequencelength200_poolinglast_returnallFalse	0.3367	0.3367	0	0
transformer_20240814-061805_d_model64_nhead4_layers2_ff256_output3_batch32_dropout0.2_sequencelength200_poolinglast_returnallTrue	0.3193	0.3193	0	0

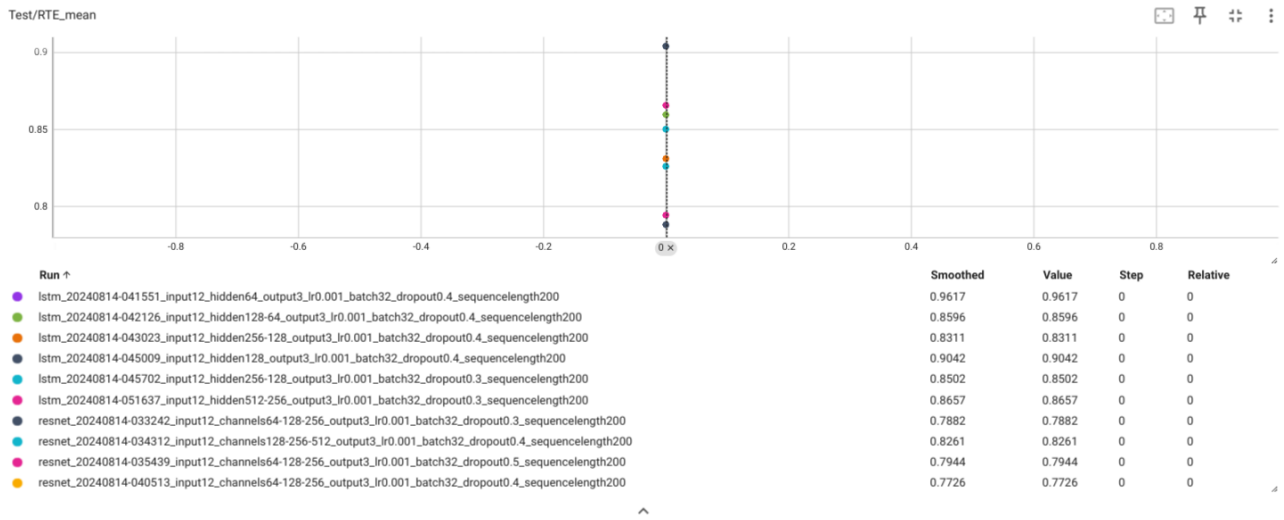
Test/MSE



Run	Smoothed	Value	Step	Relative
resnet_20240814-040513_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2126	0.2126	0	0
resnet_20240814-033242_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.2242	0.2242	0	0
resnet_20240814-035439_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.2276	0.2276	0	0
resnet_20240814-034312_input12_channels128-256-512_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2388	0.2388	0	0
lstm_20240814-043023_input12_hidden256-128_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2448	0.2448	0	0
lstm_20240814-045702_input12_hidden256-128_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.2618	0.2618	0	0
lstm_20240814-042126_input12_hidden128-64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2636	0.2636	0	0
lstm_20240814-051637_input12_hidden512-256_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.265	0.265	0	0
lstm_20240814-045009_input12_hidden128_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2854	0.2854	0	0
transformer_20240814-061805_d_model64_nhead4_layers2_ff256_output3_batch32_dropout0.2_sequencelength200_poolinglast_returnallTrue	0.3149	0.3149	0	0
lstm_20240814-041551_input12_hidden64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3247	0.3247	0	0
transformer_20240814-061004_d_model64_nhead4_layers2_ff256_output3_batch32_dropout0.2_sequencelength200_poolinglast_returnallFalse	0.3346	0.3346	0	0
transformer_20240814-055924_d_model128_nhead8_layers2_ff1024_output3_batch32_dropout0.3_sequencelength200_poolinglast_returnallFalse	0.3613	0.3613	0	0



Run	Smoothed	Value	Step	Relative
resnet_20240814-040513_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.6265	0.6265	0	0
resnet_20240814-033242_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.6389	0.6389	0	0
resnet_20240814-035439_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.6501	0.6501	0	0
lstm_20240814-043023_input12_hidden256-128_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.6618	0.6618	0	0
lstm_20240814-045702_input12_hidden256-128_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.6645	0.6645	0	0
lstm_20240814-051637_input12_hidden512-256_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.6696	0.6696	0	0
resnet_20240814-034312_input12_channels128-256-512_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.6838	0.6838	0	0
lstm_20240814-042126_input12_hidden128-64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.686	0.686	0	0
lstm_20240814-045009_input12_hidden128_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.7406	0.7406	0	0
lstm_20240814-041551_input12_hidden64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.8047	0.8047	0	0



Run	Smoothed	Value	Step	Relative
lstm_20240814-041551_input12_hidden64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.9617	0.9617	0	0
lstm_20240814-042126_input12_hidden128-64_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.8596	0.8596	0	0
lstm_20240814-043023_input12_hidden256-128_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.8311	0.8311	0	0
lstm_20240814-045009_input12_hidden128_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.9042	0.9042	0	0
lstm_20240814-045702_input12_hidden256-128_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.8502	0.8502	0	0
lstm_20240814-051637_input12_hidden512-256_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.8657	0.8657	0	0
resnet_20240814-033242_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.7882	0.7882	0	0
resnet_20240814-034312_input12_channels128-256-512_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.8261	0.8261	0	0
resnet_20240814-035439_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.7944	0.7944	0	0
resnet_20240814-040513_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.7726	0.7726	0	0

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.

Remove Parts of IMU

Removing the attitudes make the result better a little bit. Removing the derivation of attitudes(rotation rate) however, makes the result much worse.

More Epochs

Somehow if I use more epochs it still improve a lot? Although the Val Loss isn't decreasing by a lot though.

Add LayerNorm to LSTM

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 [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
```

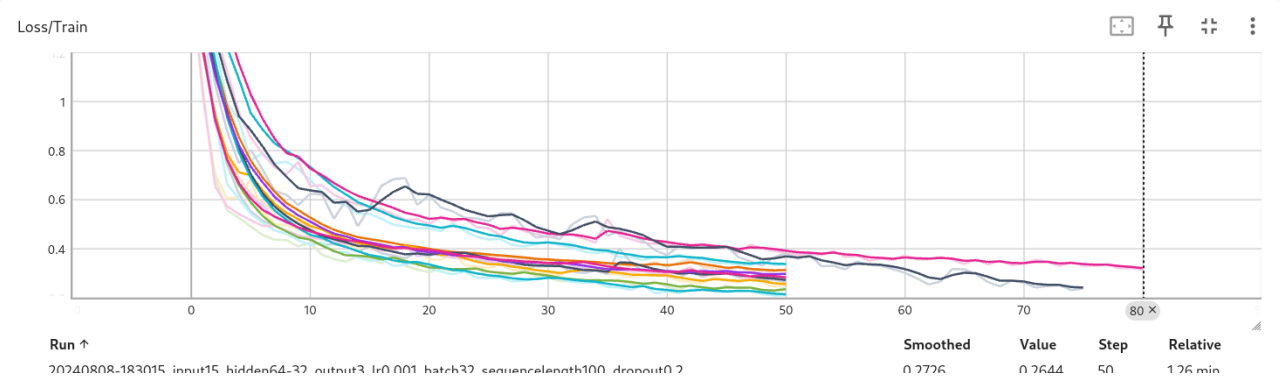
Finetuning

We need to finetune these

Basically I run these to test

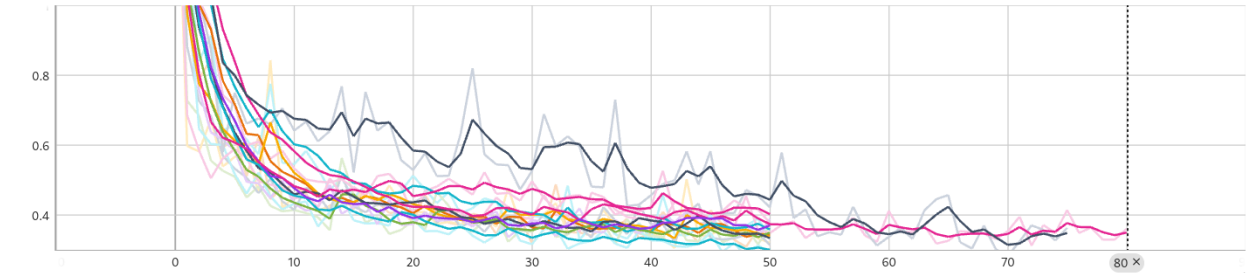
```
# Basic configuration:
python lstm_train.py --sequence_length 100 --hidden_sizes 64 32 --num_epocl
# Longer sequence length
python lstm_train.py --sequence_length 200 --hidden_sizes 64 32 --num_epocl
# Shorter sequence length
python lstm_train.py --sequence_length 50 --hidden_sizes 64 32 --num_epochs
# 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_ep
# Larger hidden sizes
python lstm_train.py --sequence_length 100 --hidden_sizes 128 64 --num_epoc
# Smaller hidden sizes
python lstm_train.py --sequence_length 100 --hidden_sizes 32 16 --num_epocl
# Complex configuration
python lstm_train.py --sequence_length 300 --hidden_sizes 128 64 32 16 --nu
###### With higher dropout #####
# Longer sequence length
python lstm_train.py --sequence_length 200 --hidden_sizes 64 32 --num_epocl

# Single layer LSTM
python lstm_train.py --sequence_length 200 --hidden_sizes 64 --num_epochs 5
```

Tensorboard Results



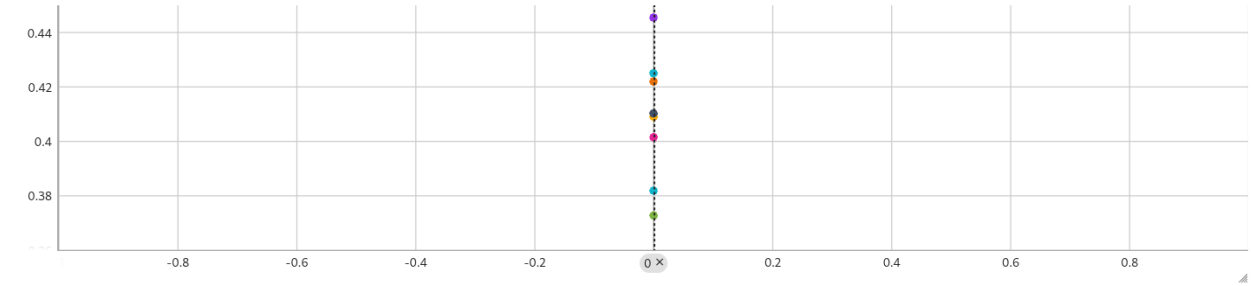
20240808-183257_input15_hidden64-32_output3_lr0.001_batch32_sequencelength200_dropout0.2	0.2141	0.2097	50	1.066 min
20240808-183609_input15_hidden64-32_output3_lr0.001_batch32__sequencelength50_dropout0.2	0.2837	0.2777	50	1.1 min
20240808-183823_input15_hidden128_output3_lr0.001_batch32_sequencelength100_dropout0.2	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



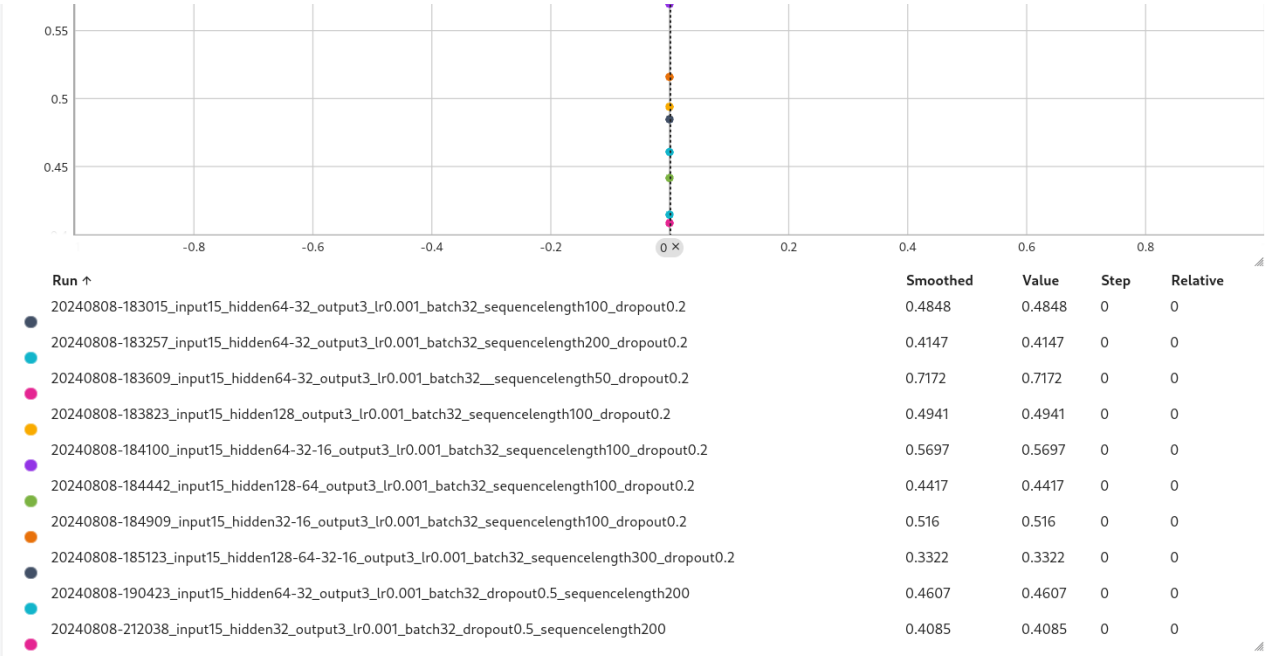
Run ↑	Smoothed	Value	Step	Relative
20240808-183015_input15_hidden64-32_output3_lr0.001_batch32_sequencelength100_dropout0.2	0.3336	0.3136	50	1.26 min
20240808-183257_input15_hidden64-32_output3_lr0.001_batch32_sequencelength200_dropout0.2	0.3003	0.2912	50	1.066 min
20240808-183609_input15_hidden64-32_output3_lr0.001_batch32__sequencelength50_dropout0.2	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

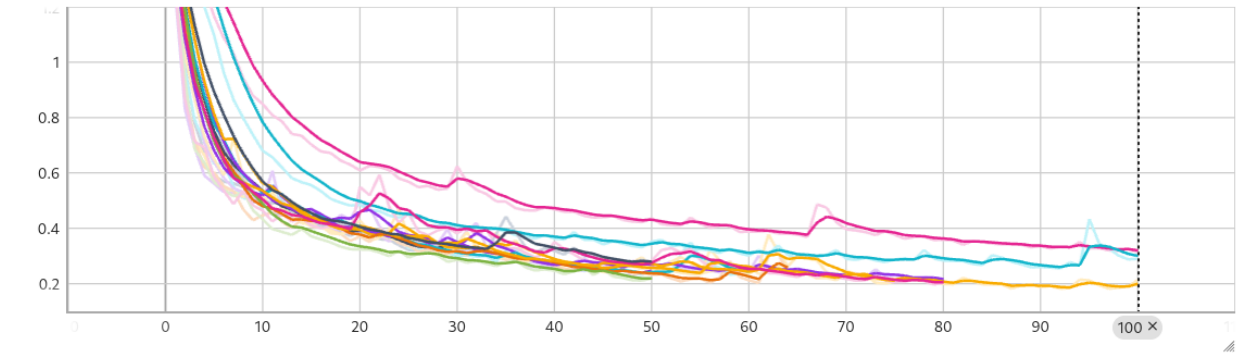


Some More Training and Finetuning

```
python lstm_train.py --sequence_length 200 --hidden_sizes 32 --num_epochs 8
python lstm_train.py --sequence_length 200 --hidden_sizes 32 --num_epochs 5
# Large Batch
python lstm_train.py --sequence_length 200 --hidden_sizes 32 --num_epochs 1
python lstm_train.py --sequence_length 200 --hidden_sizes 32 --num_epochs 1
# Large Hidden Size
python lstm_train.py --sequence_length 200 --hidden_sizes 64 --num_epochs 5

# Run it twice
python lstm_train.py --sequence_length 200 --hidden_sizes 64 --num_epochs 8
python lstm_train.py --sequence_length 200 --hidden_sizes 64 --num_epochs 8

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 8
python lstm_train.py --sequence_length 200 --hidden_sizes 128 64 --num_epoc
```

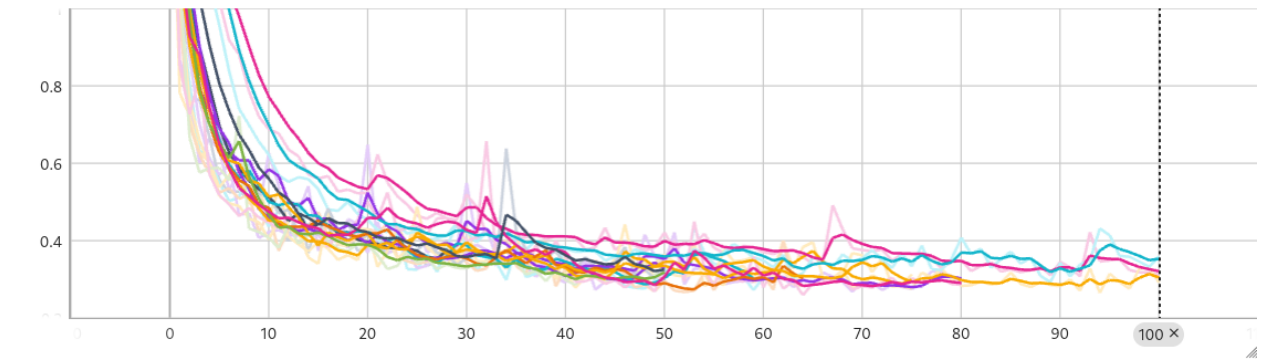
Tensorboard Results

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

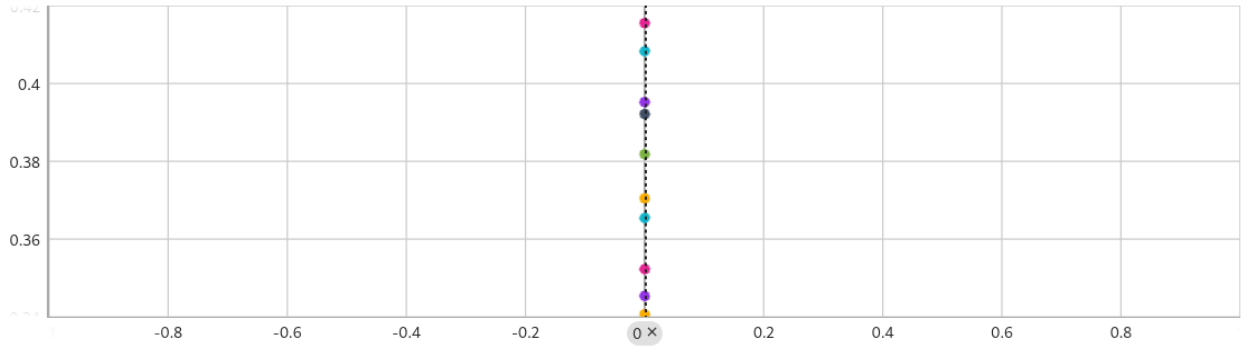
Loss/Validation



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

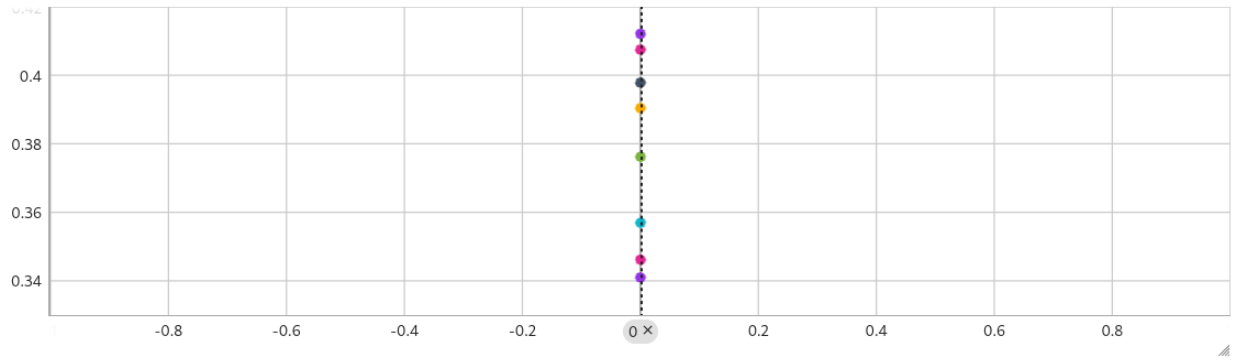
20240808-221511_input15_hidden64_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.303	0.3053	60	54.54 sec
20240808-221726_input15_hidden96_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2903	0.2857	80	2.472 min
20240808-222130_input15_hidden128-64_output3_lr0.001_batch32_dropout0.4_sequencelength200	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

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

Resnet for Localization

I removed the attitude data

Commands to run

Dummy Script

```
python resnet_train.py --num_epochs 50
```

```
# Baseline configuration
python resnet_train.py --sequence_length 200 --input_size 12 --channels 64

# Dropout Increase
python resnet_train.py --sequence_length 200 --input_size 12 --channels 64

# Longer sequence length
python resnet_train.py --sequence_length 300 --input_size 12 --channels 64

# Larger batch size
python resnet_train.py --sequence_length 200 --input_size 12 --channels 64

# Smaller batch size
```

```
python resnet_train.py --sequence_length 200 --input_size 12 --channels 64

# Higher dropout rate
python resnet_train.py --sequence_length 200 --input_size 12 --channels 64

# Deeper network
python resnet_train.py --sequence_length 200 --input_size 12 --channels 64

# Wider network
python resnet_train.py --sequence_length 200 --input_size 12 --channels 128

# Shorter sequence length with larger batch size
python resnet_train.py --sequence_length 150 --input_size 12 --channels 64

# Longer sequence length with smaller learning rate
python resnet_train.py --sequence_length 400 --input_size 12 --channels 64

# Shallow network with higher dropout
python resnet_train.py --sequence_length 200 --input_size 12 --channels 128
```

ResNet-like 1D CNN Architecture

Input Block

- BatchNorm1d (input_size channels)
- 1D Conv (input_size -> channels[0], kernel_size 7, stride 2)
- BatchNorm1d
- ReLU
- 1D MaxPool (kernel_size 3, stride 2)

Residual Blocks

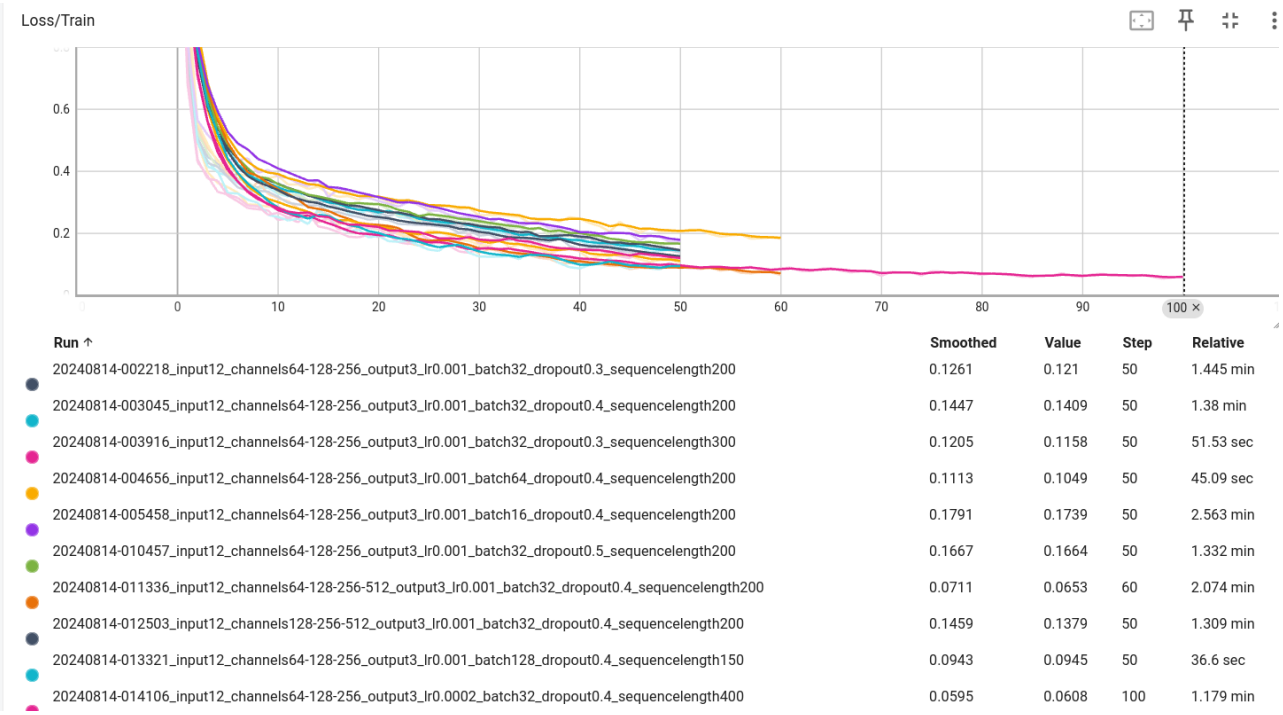
Multiple layers of ResidualBlocks, with increasing channel sizes:

- Each layer contains 2 ResidualBlocks
- ResidualBlock structure:
  1. 1D Conv (in_channels -> out_channels, kernel_size 3, stride 1 or 2)
  2. BatchNorm1d
  3. ReLU
  4. 1D Conv (out_channels -> out_channels, kernel_size 3, stride 1)
  5. BatchNorm1d
  6. Skip connection (with 1x1 Conv if needed)
  7. ReLU

Output Block

- Adaptive Average Pooling (to 1)
- Flatten
- Dropout
- Fully Connected layer (channels[-1] -> output_size)

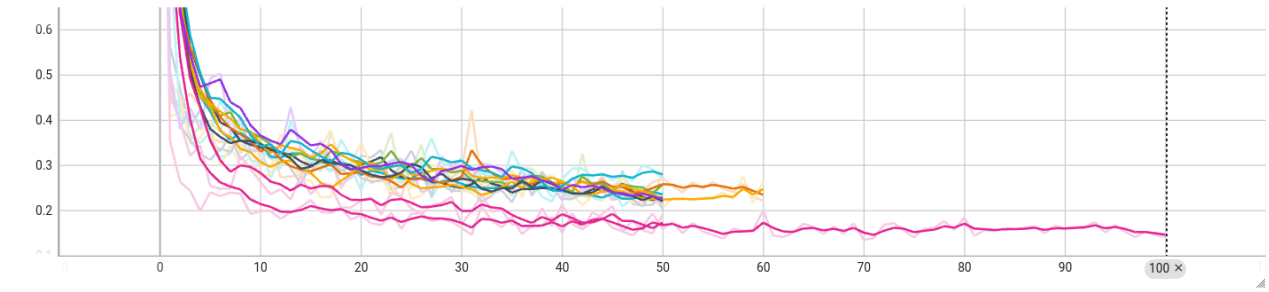
Results



Run	Smoothed	Value	Step	Relative
20240814-002218_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.1261	0.121	50	1.445 min
20240814-003045_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.1447	0.1409	50	1.38 min
20240814-003916_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.3_sequencelength300	0.1205	0.1158	50	51.53 sec
20240814-004656_input12_channels64-128-256_output3_lr0.001_batch64_dropout0.4_sequencelength200	0.1113	0.1049	50	45.09 sec
20240814-005458_input12_channels64-128-256_output3_lr0.001_batch16_dropout0.4_sequencelength200	0.1791	0.1739	50	2.563 min
20240814-010457_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.1667	0.1664	50	1.332 min
20240814-011336_input12_channels64-128-256-512_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.0711	0.0653	60	2.074 min
20240814-012503_input12_channels128-256-512_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.1459	0.1379	50	1.309 min
20240814-013321_input12_channels64-128-256_output3_lr0.001_batch128_dropout0.4_sequencelength150	0.0943	0.0945	50	36.6 sec
20240814-014106_input12_channels64-128-256_output3_lr0.0002_batch32_dropout0.4_sequencelength400	0.0595	0.0608	100	1.179 min

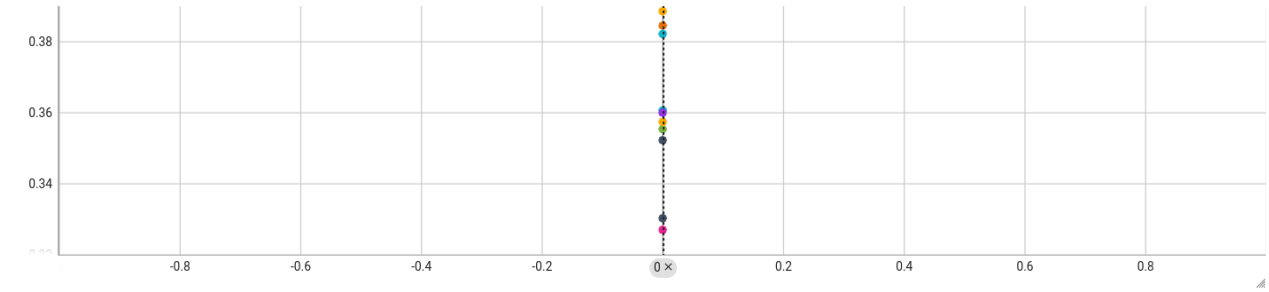
20240814-015005_input12_channels128-256_output3_lr0.001_batch32_dropout0.6_sequencelength200	0.1855	0.1827	60	1.211 min

Loss/Validation



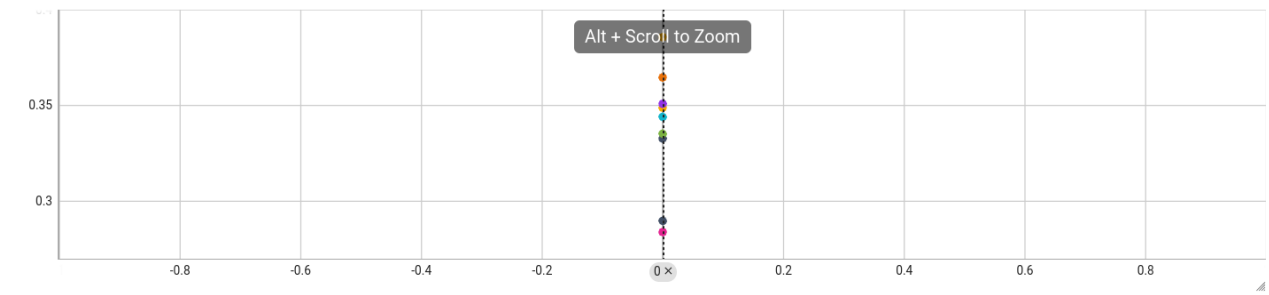
Run ↑	Smoothed	Value	Step	Relative
20240814-002218_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.2203	0.2037	50	1.445 min
20240814-003045_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2365	0.2303	50	1.38 min
20240814-003916_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.3_sequencelength300	0.1732	0.1913	50	51.53 sec
20240814-004656_input12_channels64-128-256_output3_lr0.001_batch64_dropout0.4_sequencelength200	0.2249	0.2245	50	45.09 sec
20240814-005458_input12_channels64-128-256_output3_lr0.001_batch16_dropout0.4_sequencelength200	0.2258	0.2166	50	2.563 min
20240814-010457_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.2575	0.2866	50	1.332 min
20240814-011336_input12_channels64-128-256-512_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2349	0.2219	60	2.074 min
20240814-012503_input12_channels128-256-512_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.2289	0.2375	50	1.309 min
20240814-013321_input12_channels64-128-256_output3_lr0.001_batch128_dropout0.4_sequencelength150	0.2802	0.2703	50	36.6 sec
20240814-014106_input12_channels64-128-256_output3_lr0.0002_batch32_dropout0.4_sequencelength400	0.1468	0.1421	100	1.179 min
20240814-015005_input12_channels128-256_output3_lr0.001_batch32_dropout0.6_sequencelength200	0.2475	0.2648	60	1.211 min

Test/MAE



Run ↑	Smoothed	Value	Step	Relative
20240814-002218_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.3303	0.3303	0	0
20240814-003045_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3607	0.3607	0	0
20240814-003916_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.3_sequencelength300	0.327	0.327	0	0
20240814-004656_input12_channels64-128-256_output3_lr0.001_batch64_dropout0.4_sequencelength200	0.3886	0.3886	0	0
20240814-005458_input12_channels64-128-256_output3_lr0.001_batch16_dropout0.4_sequencelength200	0.36	0.36	0	0
20240814-010457_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.3554	0.3554	0	0
20240814-011336_input12_channels64-128-256-512_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3846	0.3846	0	0
20240814-012503_input12_channels128-256-512_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3523	0.3523	0	0
20240814-013321_input12_channels64-128-256_output3_lr0.001_batch128_dropout0.4_sequencelength150	0.3822	0.3822	0	0
20240814-014106_input12_channels64-128-256_output3_lr0.0002_batch32_dropout0.4_sequencelength400	0.2987	0.2987	0	0
20240814-015005_input12_channels128-256_output3_lr0.001_batch32_dropout0.6_sequencelength200	0.3574	0.3574	0	0

Test/MSE



Run	Smoothed	Value	Step	Relative
20240814-002218_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.3_sequencelength200	0.2898	0.2898	0	0
20240814-003045_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.344	0.344	0	0
20240814-003916_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.3_sequencelength300	0.2839	0.2839	0	0
20240814-004656_input12_channels64-128-256_output3_lr0.001_batch64_dropout0.4_sequencelength200	0.3857	0.3857	0	0
20240814-005458_input12_channels64-128-256_output3_lr0.001_batch16_dropout0.4_sequencelength200	0.3509	0.3509	0	0
20240814-010457_input12_channels64-128-256_output3_lr0.001_batch32_dropout0.5_sequencelength200	0.3352	0.3352	0	0
20240814-011336_input12_channels64-128-256-512_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3646	0.3646	0	0
20240814-012503_input12_channels128-256-512_output3_lr0.001_batch32_dropout0.4_sequencelength200	0.3328	0.3328	0	0
20240814-013321_input12_channels64-128-256_output3_lr0.001_batch128_dropout0.4_sequencelength150	0.4031	0.4031	0	0
20240814-014106_input12_channels64-128-256_output3_lr0.0002_batch32_dropout0.4_sequencelength400	0.2186	0.2186	0	0
20240814-015005_input12_channels128-256_output3_lr0.001_batch32_dropout0.6_sequencelength200	0.3489	0.3489	0	0

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).

Performance

It's like transfomer generally is better than LSTM(at least in my implementation), but not by a lot, it suffers much worse from overfitting.

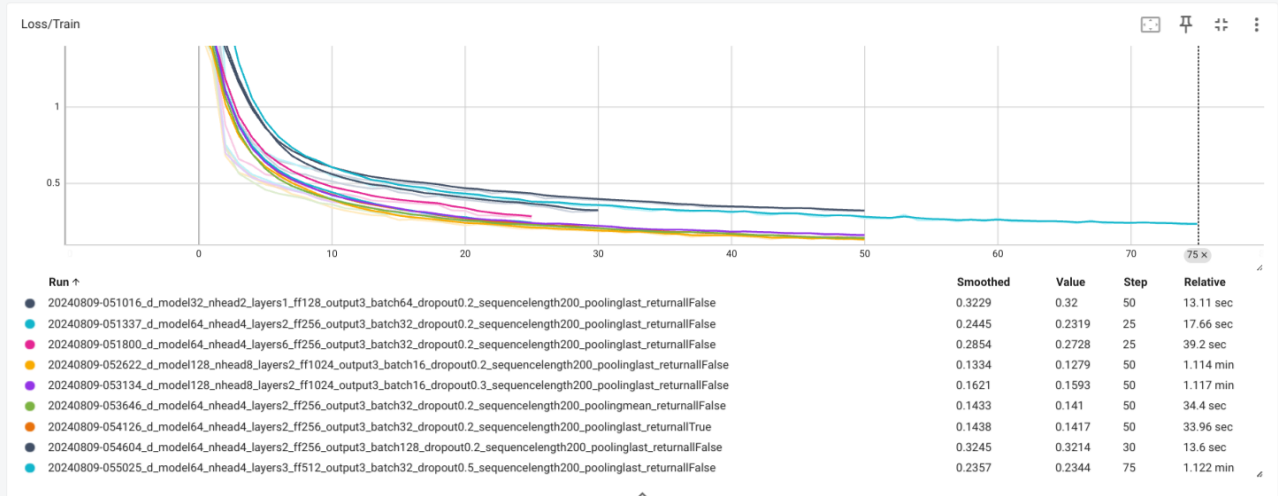
Run Some Configuarations

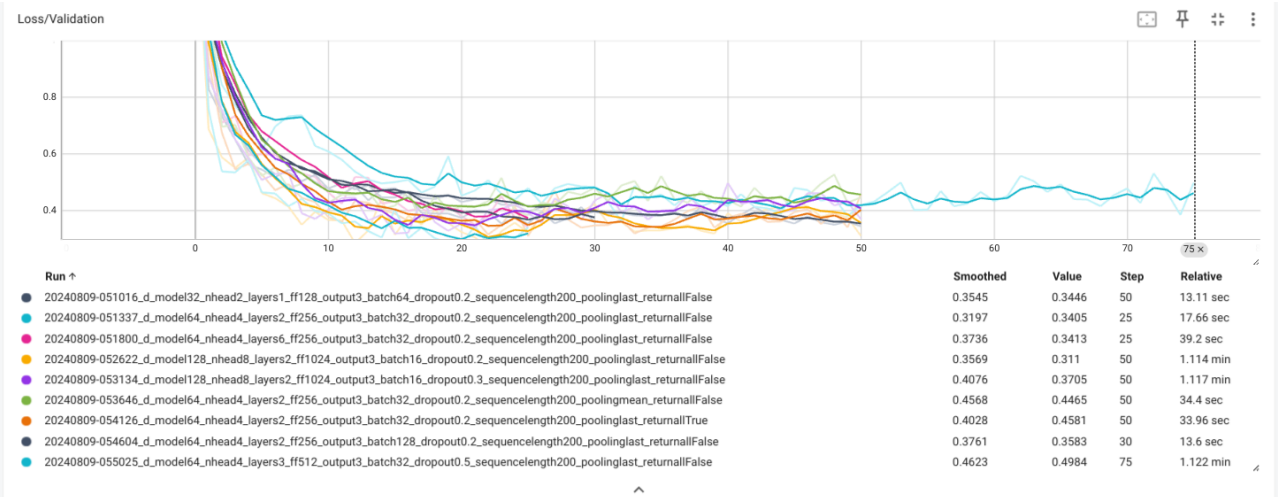
So basically run different configs to get an idea what happens

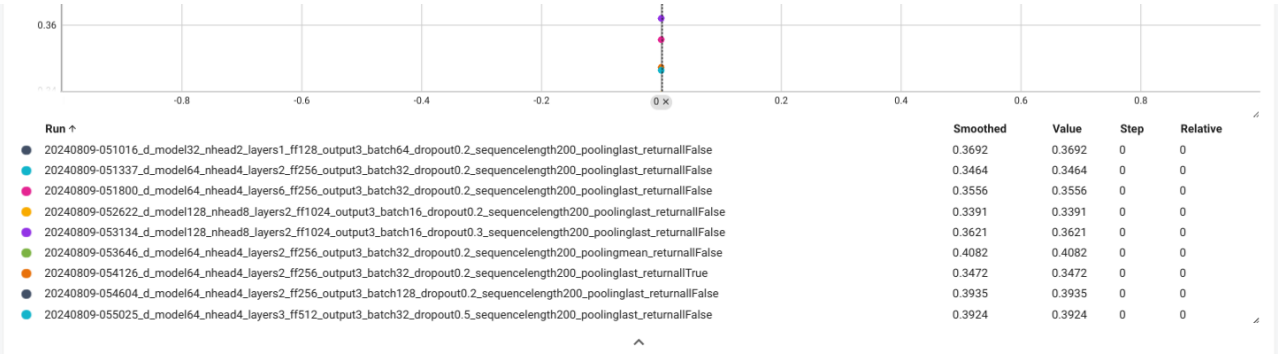
```
# Baseline configuration
python transformer_train.py --d_model 32 --nhead 2 --num_layers 1 --dim_fe

# 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
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

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

Results

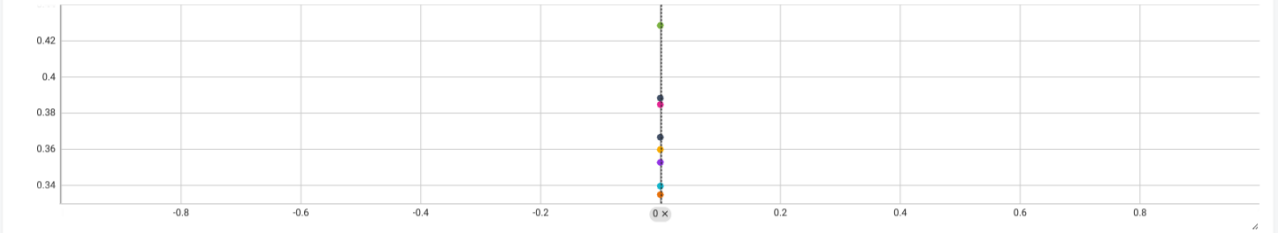






Run	Smoothed	Value	Step	Relative
20240809-051016_d_model32_nhead2_layers1_ff128_output3_batch64_dropout0.2_sequencelength200_poolinglast_returnallFalse	0.3692	0.3692	0	0
20240809-051337_d_model64_nhead4_layers2_ff256_output3_batch32_dropout0.2_sequencelength200_poolinglast_returnallFalse	0.3464	0.3464	0	0
20240809-051800_d_model64_nhead4_layers6_ff256_output3_batch32_dropout0.2_sequencelength200_poolinglast_returnallFalse	0.3556	0.3556	0	0
20240809-052622_d_model128_nhead8_layers2_ff1024_output3_batch16_dropout0.2_sequencelength200_poolinglast_returnallFalse	0.3391	0.3391	0	0
20240809-053134_d_model128_nhead8_layers2_ff1024_output3_batch16_dropout0.3_sequencelength200_poolinglast_returnallFalse	0.3621	0.3621	0	0
20240809-053646_d_model64_nhead4_layers2_ff256_output3_batch32_dropout0.2_sequencelength200_poolingmean_returnallFalse	0.4082	0.4082	0	0
20240809-054126_d_model64_nhead4_layers2_ff256_output3_batch32_dropout0.2_sequencelength200_poolinglast_returnallTrue	0.3472	0.3472	0	0
20240809-054604_d_model64_nhead4_layers2_ff256_output3_batch128_dropout0.2_sequencelength200_poolinglast_returnallFalse	0.3935	0.3935	0	0
20240809-055025_d_model64_nhead4_layers3_ff512_output3_batch32_dropout0.5_sequencelength200_poolinglast_returnallFalse	0.3924	0.3924	0	0

Test/MSE



Run	Smoothed	Value	Step	Relative
20240809-051016_d_model32_nhead2_layers1_ff128_output3_batch64_dropout0.2_sequencelength200_poolinglast_returnallFalse	0.3666	0.3666	0	0
20240809-051337_d_model64_nhead4_layers2_ff256_output3_batch32_dropout0.2_sequencelength200_poolinglast_returnallFalse	0.3396	0.3396	0	0
20240809-051800_d_model64_nhead4_layers6_ff256_output3_batch32_dropout0.2_sequencelength200_poolinglast_returnallFalse	0.3847	0.3847	0	0
20240809-052622_d_model128_nhead8_layers2_ff1024_output3_batch16_dropout0.2_sequencelength200_poolinglast_returnallFalse	0.3599	0.3599	0	0
20240809-053134_d_model128_nhead8_layers2_ff1024_output3_batch16_dropout0.3_sequencelength200_poolinglast_returnallFalse	0.3527	0.3527	0	0
20240809-053646_d_model64_nhead4_layers2_ff256_output3_batch32_dropout0.2_sequencelength200_poolingmean_returnallFalse	0.4286	0.4286	0	0
20240809-054126_d_model64_nhead4_layers2_ff256_output3_batch32_dropout0.2_sequencelength200_poolinglast_returnallTrue	0.3349	0.3349	0	0
20240809-054604_d_model64_nhead4_layers2_ff256_output3_batch128_dropout0.2_sequencelength200_poolinglast_returnallFalse	0.3883	0.3883	0	0
20240809-055025_d_model64_nhead4_layers3_ff512_output3_batch32_dropout0.5_sequencelength200_poolinglast_returnallFalse	0.4489	0.4489	0	0

Observations

Transformer easily achieves a much better baseline but seems to be an overkill in this problem with limited data. It have a huge variance and the result is on par with LSTM.