DLOps Assignment 2

Question 1:

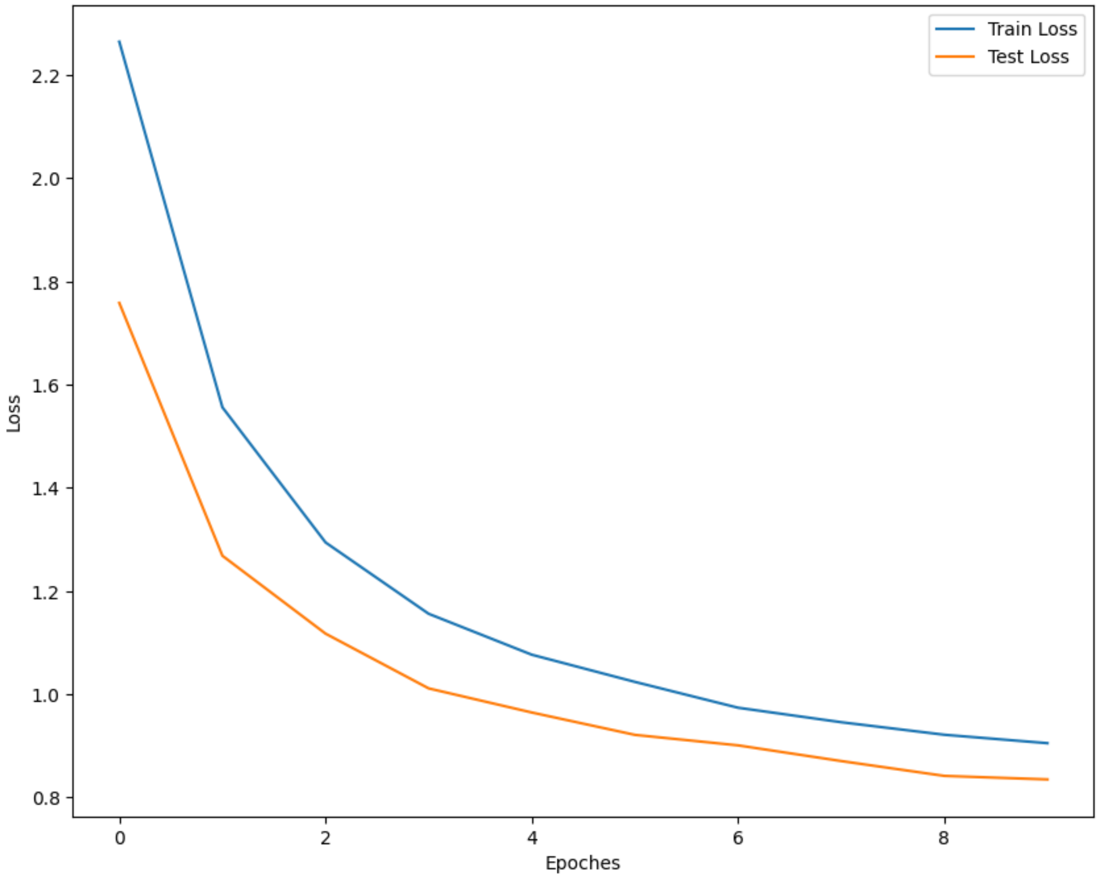
- Import require libraries
- Write device agnostic code
- Load dataset
- Preprocess and tokenize it
- Built iterator of data
- Write train and test loop for model
- Plot function for loss
- Build model for LSTM Seq2Seq, RNN Seq2Seq
- Tune hyperparameter on it
- Compare LSTM Seq2Seq and RNN Seq2Seq
- Build Attention based LSTM Seq2Seq model
- Now train all model

A

LSTM:

```
Epoch: 1 / 10
Train Loss: 2.2647 | Test Loss: 1.7585

Epoch: 2 / 10
Train Loss: 1.5561 | Test Loss: 1.2685

Epoch: 3 / 10
Train Loss: 1.2943 | Test Loss: 1.1176

Epoch: 4 / 10
Train Loss: 1.1560 | Test Loss: 1.0115

Epoch: 5 / 10
Train Loss: 1.0767 | Test Loss: 0.9648

Epoch: 6 / 10
Train Loss: 1.0240 | Test Loss: 0.9214

Epoch: 7 / 10
Train Loss: 0.9740 | Test Loss: 0.9011

Epoch: 8 / 10
Train Loss: 0.9459 | Test Loss: 0.8707
```

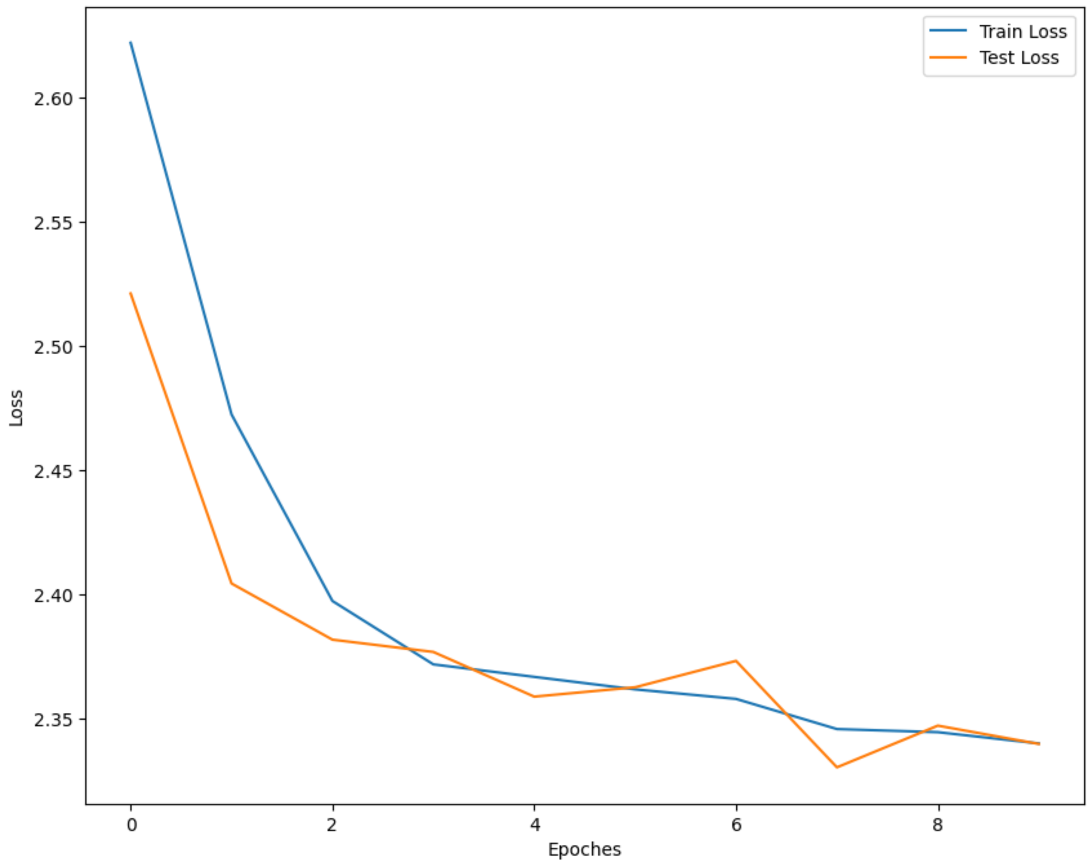
```
Epoch: 9 / 10
Train Loss: 0.9216 | Test Loss: 0.8419

Epoch: 10 / 10
Train Loss: 0.9053 | Test Loss: 0.8350

Total Exe Time: 371.58694307299993 Sec
```



RNN

```
Epoch: 1 / 10
Train Loss: 2.6220 | Test Loss: 2.5211

Epoch: 2 / 10
Train Loss: 2.4725 | Test Loss: 2.4045

Epoch: 3 / 10
Train Loss: 2.3974 | Test Loss: 2.3819

Epoch: 4 / 10
Train Loss: 2.3719 | Test Loss: 2.3769
```

```
Epoch: 5 / 10
Train Loss: 2.3669 | Test Loss: 2.3589

Epoch: 6 / 10
Train Loss: 2.3619 | Test Loss: 2.3627

Epoch: 7 / 10
Train Loss: 2.3580 | Test Loss: 2.3733

Epoch: 8 / 10
Train Loss: 2.3459 | Test Loss: 2.3305

Epoch: 9 / 10
Train Loss: 2.3446 | Test Loss: 2.3473

Epoch: 10 / 10
Train Loss: 2.3401 | Test Loss: 2.3399

Total Exe Time: 179.6433392459985 Sec
```



Observation:

- LSTM S2S loss decreased very well from 2.26 to 0.90 (train) and (test) from 1.75 to 0.83.
- Meanwhile RNN S2S loss decreased from 2.62 to 2.34 (train) and (test) from 2.52 to 2.34.
- Thus, the LSTM S2S model learns much better than RNN S2S BUt LSTM take more time than RNN.

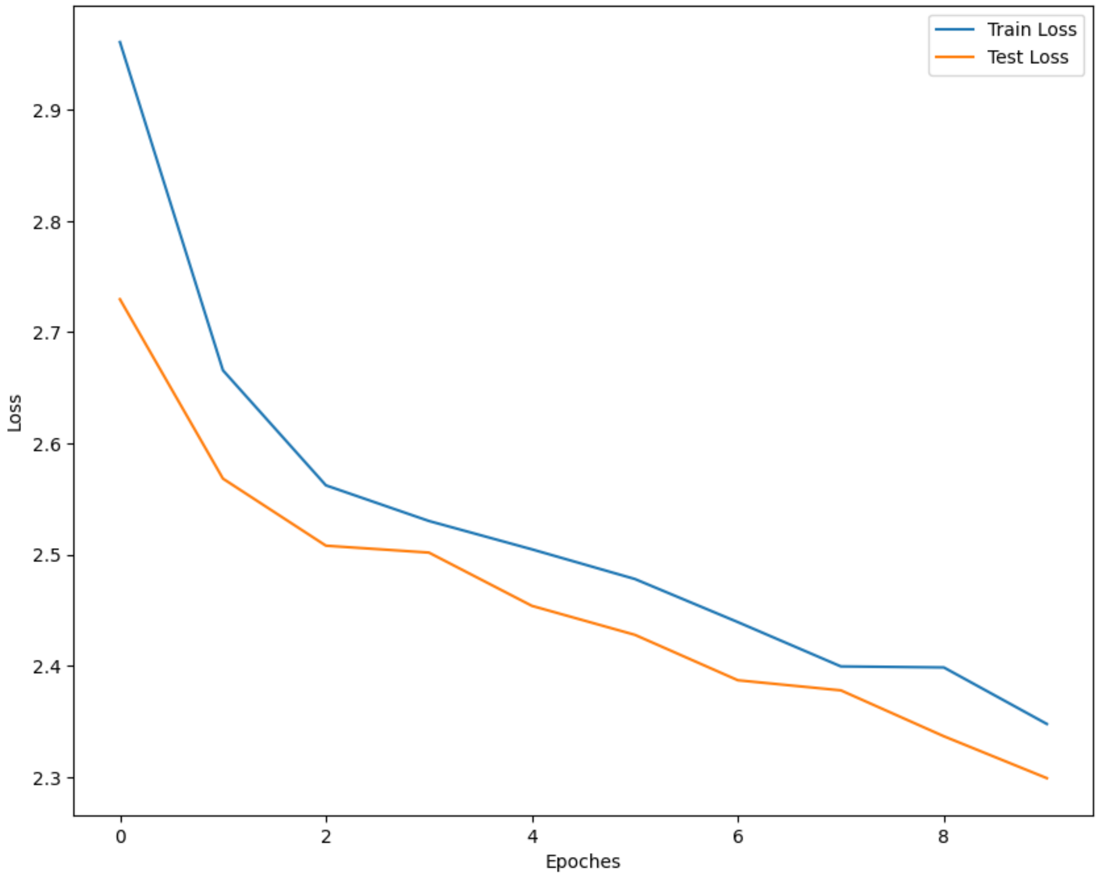
B

NOTE: for 3 number of encoder and decoder system ram crashed so there for I avoid number 3 for encoder and decoder

LSTM

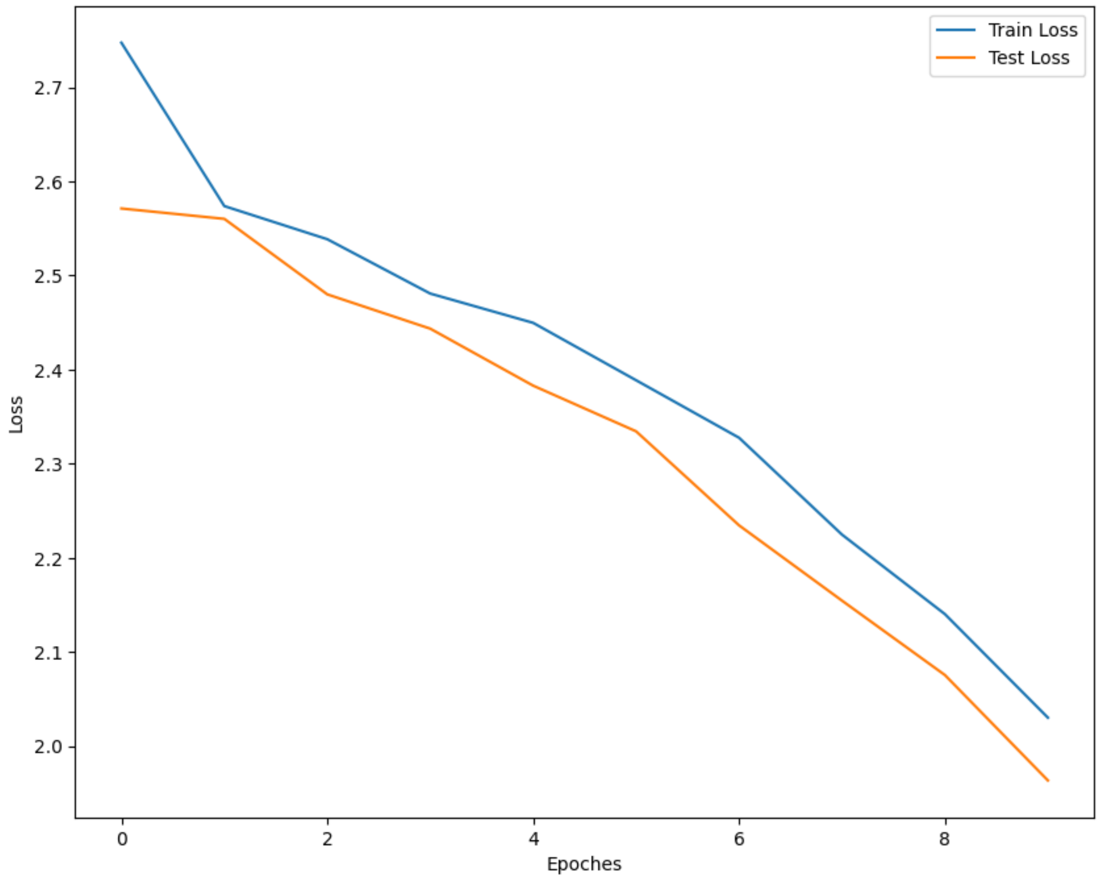
```
iter: 1 / 4
Tuning with --- encoder embedding size: 16, decoder embedding size: 16, hidden size: 16

Epoch: 1 / 10
Train Loss: 2.9608 | Test Loss: 2.7296

Epoch: 2 / 10
Train Loss: 2.6657 | Test Loss: 2.5683

Epoch: 3 / 10
Train Loss: 2.5622 | Test Loss: 2.5080

Epoch: 4 / 10
Train Loss: 2.5302 | Test Loss: 2.5018

Epoch: 5 / 10
Train Loss: 2.5047 | Test Loss: 2.4539

Epoch: 6 / 10
Train Loss: 2.4780 | Test Loss: 2.4279

Epoch: 7 / 10
Train Loss: 2.4393 | Test Loss: 2.3870

Epoch: 8 / 10
Train Loss: 2.3995 | Test Loss: 2.3779

Epoch: 9 / 10
Train Loss: 2.3985 | Test Loss: 2.3367
```

```
Epoch: 10 / 10
Train Loss: 2.3477 | Test Loss: 2.2990

Total Exe Time: 25.1321322770018 Sec
```



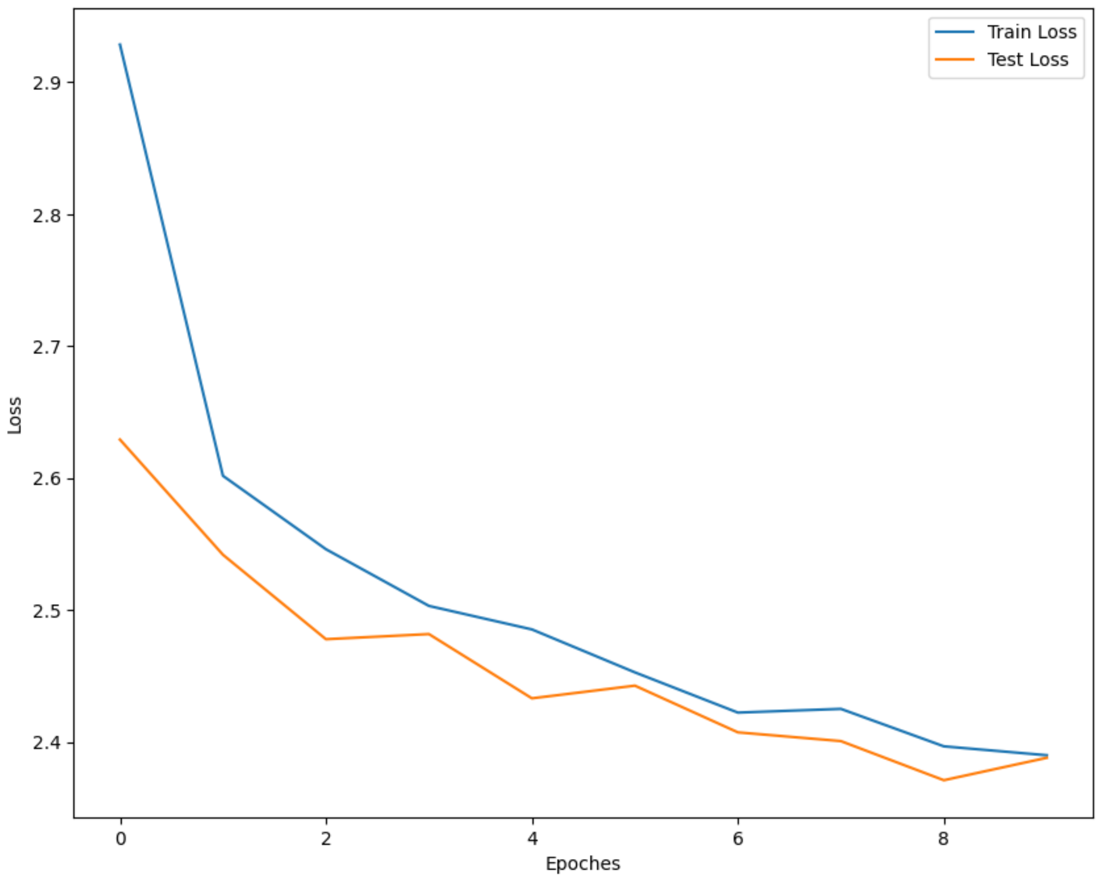
```
iter: 2 / 4
Tuning with --- encoder embedding size: 16, decoder embedding size: 16, hidden size: 64

Epoch: 1 / 10
Train Loss: 2.7475 | Test Loss: 2.5713

Epoch: 2 / 10
Train Loss: 2.5738 | Test Loss: 2.5602

Epoch: 3 / 10
Train Loss: 2.5386 | Test Loss: 2.4800

Epoch: 4 / 10
Train Loss: 2.4808 | Test Loss: 2.4436
```

```
Epoch: 5 / 10
Train Loss: 2.4497 | Test Loss: 2.3830

Epoch: 6 / 10
Train Loss: 2.3887 | Test Loss: 2.3344

Epoch: 7 / 10
Train Loss: 2.3276 | Test Loss: 2.2345

Epoch: 8 / 10
Train Loss: 2.2247 | Test Loss: 2.1545

Epoch: 9 / 10
Train Loss: 2.1402 | Test Loss: 2.0754

Epoch: 10 / 10
Train Loss: 2.0301 | Test Loss: 1.9634

Total Exe Time: 36.38090182199812 Sec
```



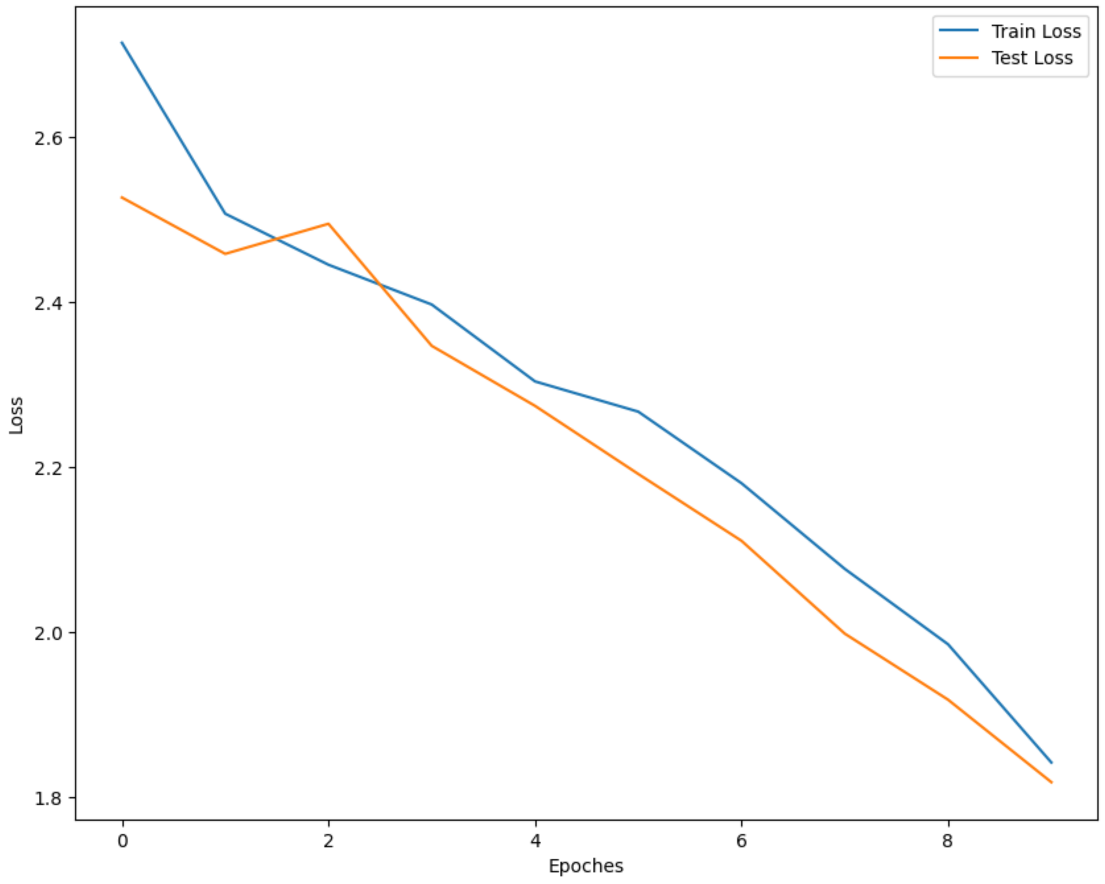
```
iter: 3 / 4
```

```
Tuning with --- encoder embedding size: 64, decoder embedding size: 64, hidden size: 16

Epoch: 1 / 10
Train Loss: 2.9284 | Test Loss: 2.6291

Epoch: 2 / 10
Train Loss: 2.6017 | Test Loss: 2.5419

Epoch: 3 / 10
Train Loss: 2.5460 | Test Loss: 2.4779

Epoch: 4 / 10
Train Loss: 2.5031 | Test Loss: 2.4818

Epoch: 5 / 10
Train Loss: 2.4853 | Test Loss: 2.4331

Epoch: 6 / 10
Train Loss: 2.4527 | Test Loss: 2.4427

Epoch: 7 / 10
Train Loss: 2.4223 | Test Loss: 2.4073

Epoch: 8 / 10
Train Loss: 2.4251 | Test Loss: 2.4007

Epoch: 9 / 10
Train Loss: 2.3967 | Test Loss: 2.3710

Epoch: 10 / 10
Train Loss: 2.3899 | Test Loss: 2.3880

Total Exe Time: 23.460940119999577 Sec
```



```
iter: 4 / 4
Tuning with --- encoder embedding size: 64, decoder embedding size: 64, hidden size: 64

Epoch: 1 / 10
Train Loss: 2.7147 | Test Loss: 2.5272

Epoch: 2 / 10
Train Loss: 2.5076 | Test Loss: 2.4590

Epoch: 3 / 10
Train Loss: 2.4458 | Test Loss: 2.4954

Epoch: 4 / 10
Train Loss: 2.3976 | Test Loss: 2.3475

Epoch: 5 / 10
Train Loss: 2.3044 | Test Loss: 2.2748

Epoch: 6 / 10
Train Loss: 2.2678 | Test Loss: 2.1922
```

```
Epoch: 7 / 10
Train Loss: 2.1810 | Test Loss: 2.1112

Epoch: 8 / 10
Train Loss: 2.0772 | Test Loss: 1.9987

Epoch: 9 / 10
Train Loss: 1.9859 | Test Loss: 1.9187

Epoch: 10 / 10
Train Loss: 1.8427 | Test Loss: 1.8188

Total Exe Time: 36.316302588002145 Sec
```

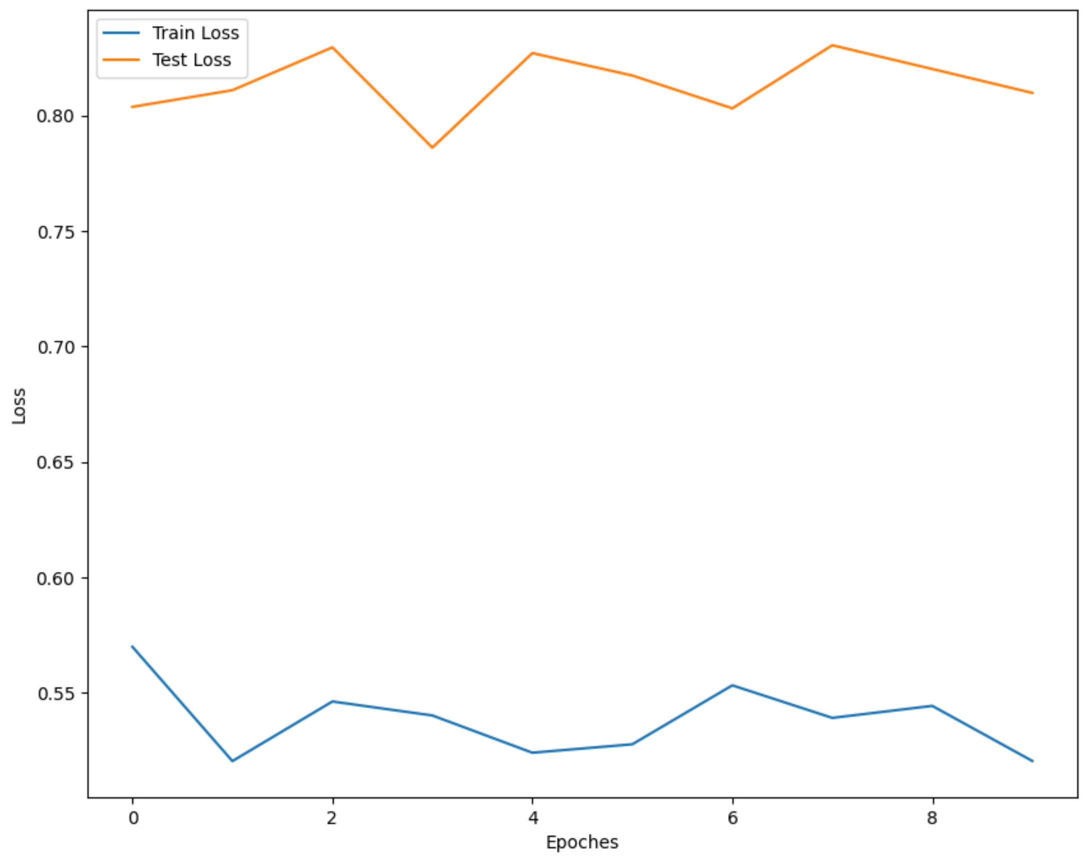


Observation:

- Best performance for LSTM at hidden node 64, encoder embedded node 64 and decoder embedded node 64 loss goes from 2.71 to 1.84 (train) and from 2.52 to 1.81 (test).

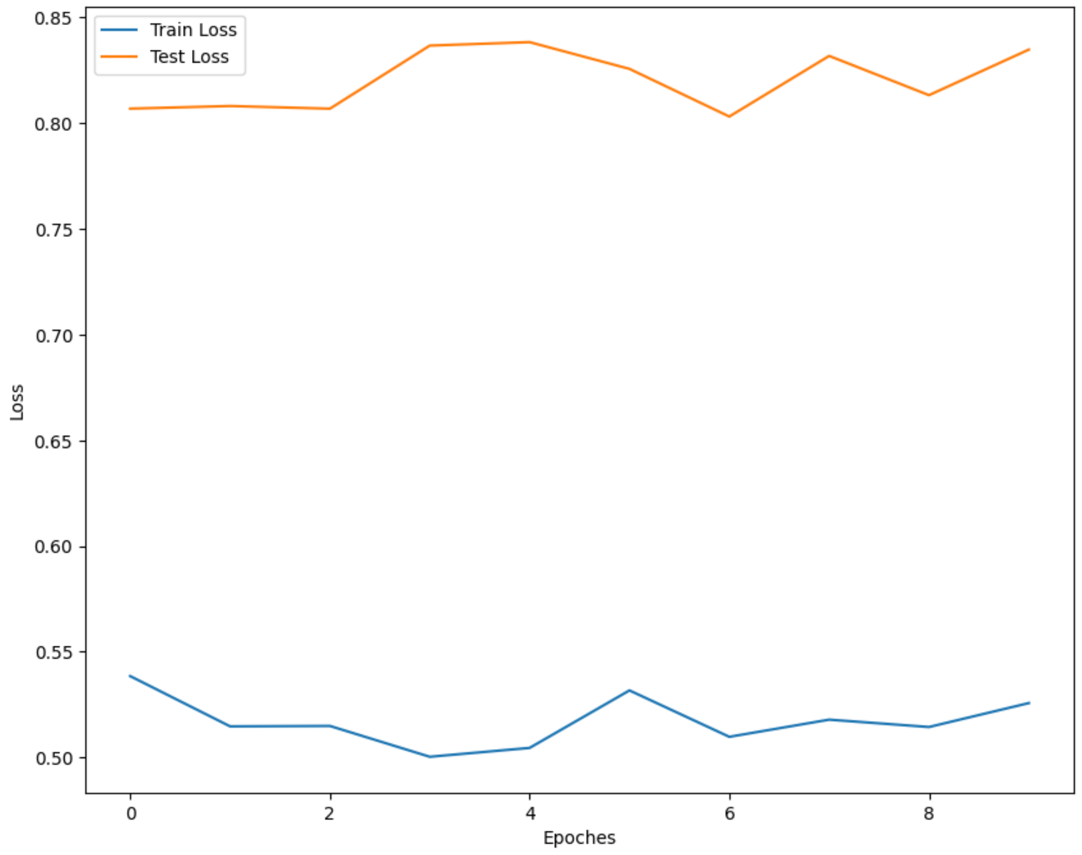
RNN

```
iter: 1 / 4
Tuning with --- encoder embedding size: 16, decoder embedding size: 16, hidden size: 16

Epoch: 1 / 10
Train Loss: 0.5700 | Test Loss: 0.8037

Epoch: 2 / 10
Train Loss: 0.5205 | Test Loss: 0.8109

Epoch: 3 / 10
Train Loss: 0.5463 | Test Loss: 0.8294

Epoch: 4 / 10
Train Loss: 0.5403 | Test Loss: 0.7860

Epoch: 5 / 10
Train Loss: 0.5242 | Test Loss: 0.8269

Epoch: 6 / 10
Train Loss: 0.5278 | Test Loss: 0.8172

Epoch: 7 / 10
Train Loss: 0.5533 | Test Loss: 0.8030

Epoch: 8 / 10
Train Loss: 0.5392 | Test Loss: 0.8303

Epoch: 9 / 10
Train Loss: 0.5444 | Test Loss: 0.8201

Epoch: 10 / 10
Train Loss: 0.5206 | Test Loss: 0.8097

Total Exe Time: 43.46197586299968 Sec
```



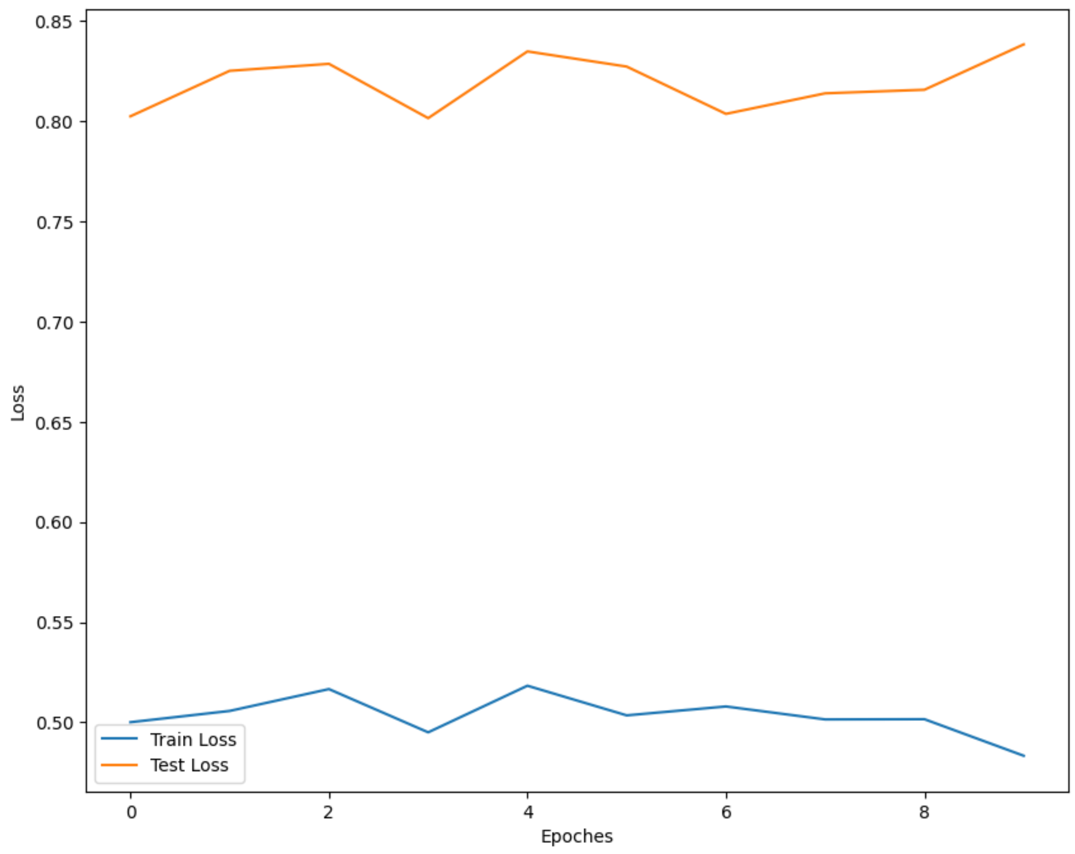
```
iter: 2 / 4
Tuning with --- encoder embedding size: 16, decoder embedding size: 16, hidden size: 64

Epoch: 1 / 10
Train Loss: 0.5385 | Test Loss: 0.8068

Epoch: 2 / 10
Train Loss: 0.5148 | Test Loss: 0.8080

Epoch: 3 / 10
Train Loss: 0.5150 | Test Loss: 0.8068

Epoch: 4 / 10
Train Loss: 0.5004 | Test Loss: 0.8366

Epoch: 5 / 10
Train Loss: 0.5046 | Test Loss: 0.8383

Epoch: 6 / 10
Train Loss: 0.5318 | Test Loss: 0.8256
```

```
Epoch: 7 / 10
Train Loss: 0.5098 | Test Loss: 0.8031

Epoch: 8 / 10
Train Loss: 0.5179 | Test Loss: 0.8317

Epoch: 9 / 10
Train Loss: 0.5145 | Test Loss: 0.8132

Epoch: 10 / 10
Train Loss: 0.5258 | Test Loss: 0.8347

Total Exe Time: 42.04093869099961 Sec
```



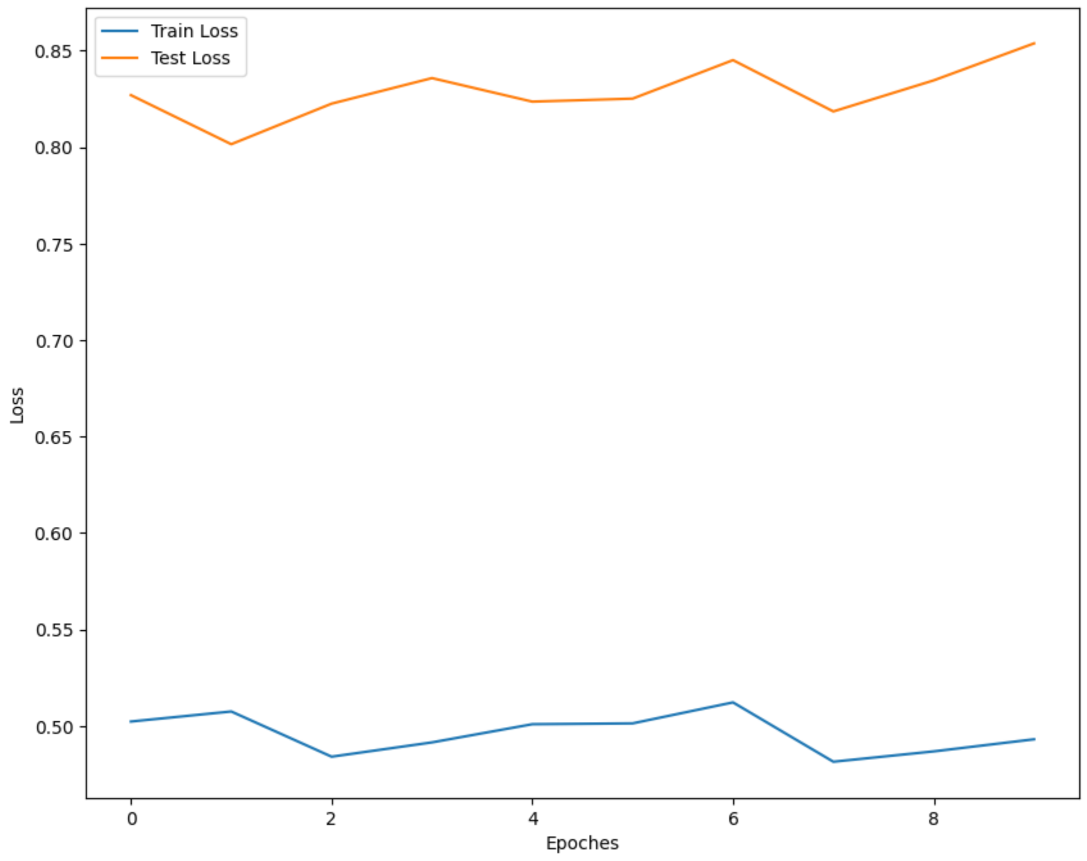
```
iter: 3 / 4
Tuning with --- encoder embedding size: 64, decoder embedding size: 64, hidden size: 16

Epoch: 1 / 10
Train Loss: 0.5001 | Test Loss: 0.8024

Epoch: 2 / 10
```

```
Train Loss: 0.5057 | Test Loss: 0.8251

Epoch: 3 / 10
Train Loss: 0.5166 | Test Loss: 0.8286

Epoch: 4 / 10
Train Loss: 0.4950 | Test Loss: 0.8015

Epoch: 5 / 10
Train Loss: 0.5183 | Test Loss: 0.8348

Epoch: 6 / 10
Train Loss: 0.5035 | Test Loss: 0.8272

Epoch: 7 / 10
Train Loss: 0.5079 | Test Loss: 0.8036

Epoch: 8 / 10
Train Loss: 0.5015 | Test Loss: 0.8138

Epoch: 9 / 10
Train Loss: 0.5016 | Test Loss: 0.8156

Epoch: 10 / 10
Train Loss: 0.4833 | Test Loss: 0.8382

Total Exe Time: 41.84626204799861 Sec
```



```
iter: 4 / 4
Tuning with --- encoder embedding size: 64, encoder embedding size: 64, hidden size: 64

Epoch: 1 / 10
Train Loss: 0.5023 | Test Loss: 0.8269

Epoch: 2 / 10
Train Loss: 0.5076 | Test Loss: 0.8015

Epoch: 3 / 10
Train Loss: 0.4841 | Test Loss: 0.8225

Epoch: 4 / 10
Train Loss: 0.4915 | Test Loss: 0.8357

Epoch: 5 / 10
Train Loss: 0.5010 | Test Loss: 0.8235

Epoch: 6 / 10
Train Loss: 0.5014 | Test Loss: 0.8251

Epoch: 7 / 10
```

```
Train Loss: 0.5123 | Test Loss: 0.8450

Epoch: 8 / 10
Train Loss: 0.4815 | Test Loss: 0.8185

Epoch: 9 / 10
Train Loss: 0.4869 | Test Loss: 0.8345

Epoch: 10 / 10
Train Loss: 0.4931 | Test Loss: 0.8537

Total Exe Time: 42.91983340799925 Sec
```



Observation:

- RNN overfit in every hyper parameter
- So LSTM is better than RNN for Seq2Seq model

C

Q 1: RNN take less time than LSTM

LSTMs are a particular kind of RNN created to solve the vanishing gradient issue that might arise in conventional RNNs. To better capture long-range dependencies in sequential data, LSTMs employ a more intricate structure with gating features. In contrast to more straightforward RNNs, LSTMs may need more training data and longer training times due to their higher complexity.

Q 2: Dropout give better output

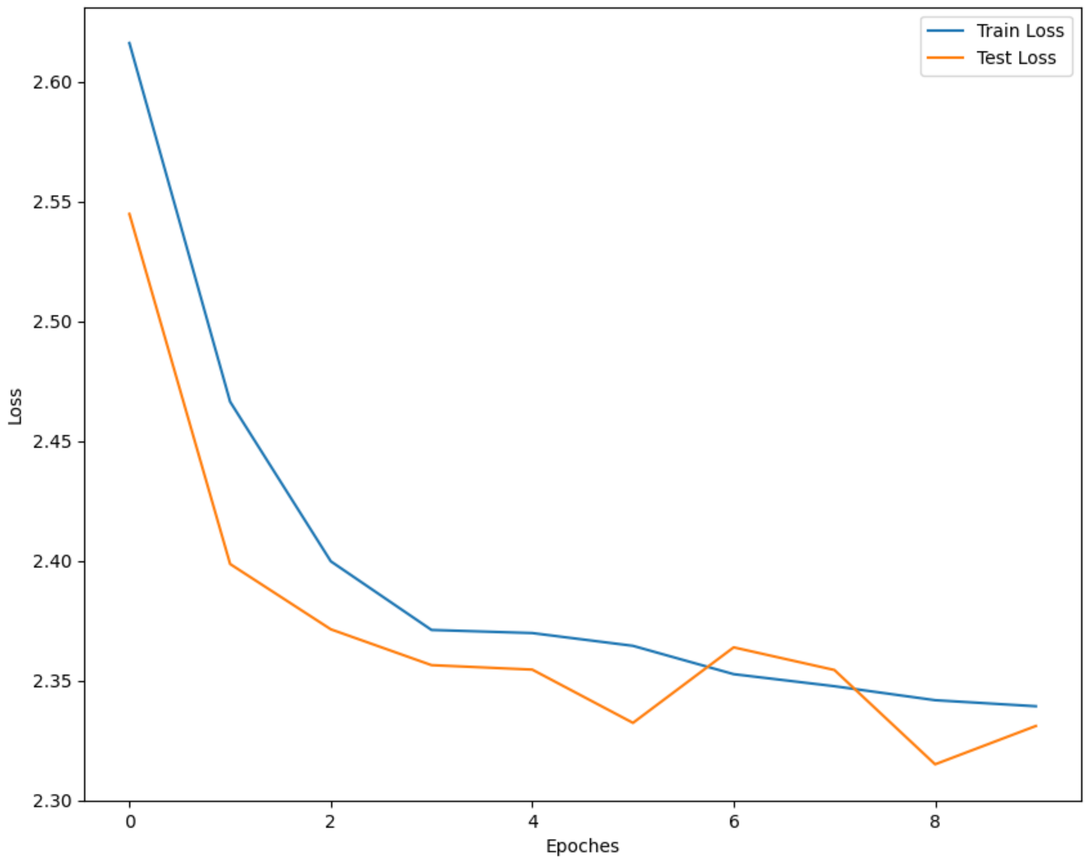
Yes it reduces the chance of overfitting

Proof: `for proof use rnn because lstm used to avoid overfitting in rnn and i use dropout in part A so here I show the result of both without dropout and with dropout but with dropout result take from part A output.`

Without Dropout

```
Epoch: 1 / 10
Train Loss: 2.6161 | Test Loss: 2.5447

Epoch: 2 / 10
Train Loss: 2.4663 | Test Loss: 2.3986

Epoch: 3 / 10
Train Loss: 2.3997 | Test Loss: 2.3713

Epoch: 4 / 10
Train Loss: 2.3710 | Test Loss: 2.3563

Epoch: 5 / 10
Train Loss: 2.3697 | Test Loss: 2.3545

Epoch: 6 / 10
Train Loss: 2.3644 | Test Loss: 2.3322

Epoch: 7 / 10
Train Loss: 2.3526 | Test Loss: 2.3638

Epoch: 8 / 10
Train Loss: 2.3476 | Test Loss: 2.3543

Epoch: 9 / 10
Train Loss: 2.3417 | Test Loss: 2.3149
```

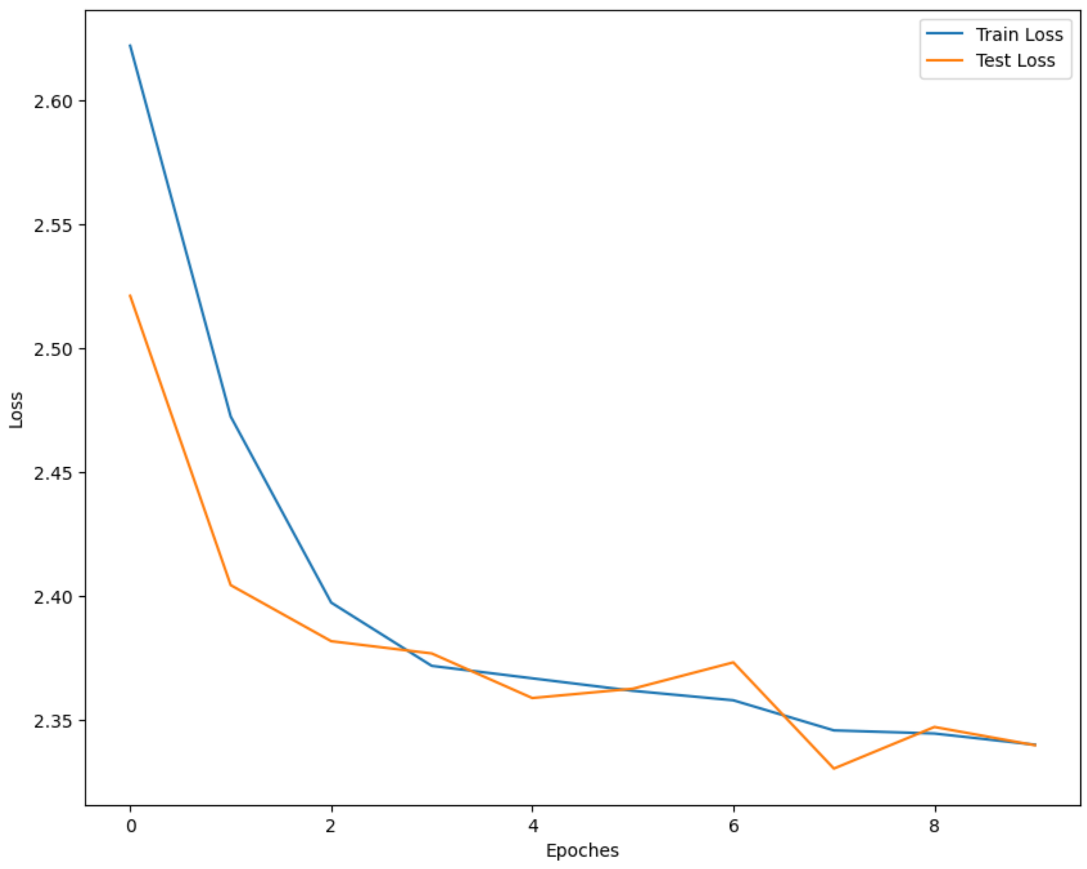
```
Epoch: 10 / 10
Train Loss: 2.3392 | Test Loss: 2.3309

Total Exe Time: 191.12200533899886 Sec
```



With Dropout

```
Epoch: 1 / 10
Train Loss: 2.6220 | Test Loss: 2.5211

Epoch: 2 / 10
Train Loss: 2.4725 | Test Loss: 2.4045

Epoch: 3 / 10
Train Loss: 2.3974 | Test Loss: 2.3819

Epoch: 4 / 10
Train Loss: 2.3719 | Test Loss: 2.3769
```

```
Epoch: 5 / 10
Train Loss: 2.3669 | Test Loss: 2.3589

Epoch: 6 / 10
Train Loss: 2.3619 | Test Loss: 2.3627

Epoch: 7 / 10
Train Loss: 2.3580 | Test Loss: 2.3733

Epoch: 8 / 10
Train Loss: 2.3459 | Test Loss: 2.3305

Epoch: 9 / 10
Train Loss: 2.3446 | Test Loss: 2.3473

Epoch: 10 / 10
Train Loss: 2.3401 | Test Loss: 2.3399

Total Exe Time: 179.6433392459985 Sec
```



Observation:

- Both loss curve are similar but at some point
- Dropout loss curve is less fluctuate than without dropout one.
- It shows the sign of overfitting

Q 3:

LSTM:

Hidden size : 8

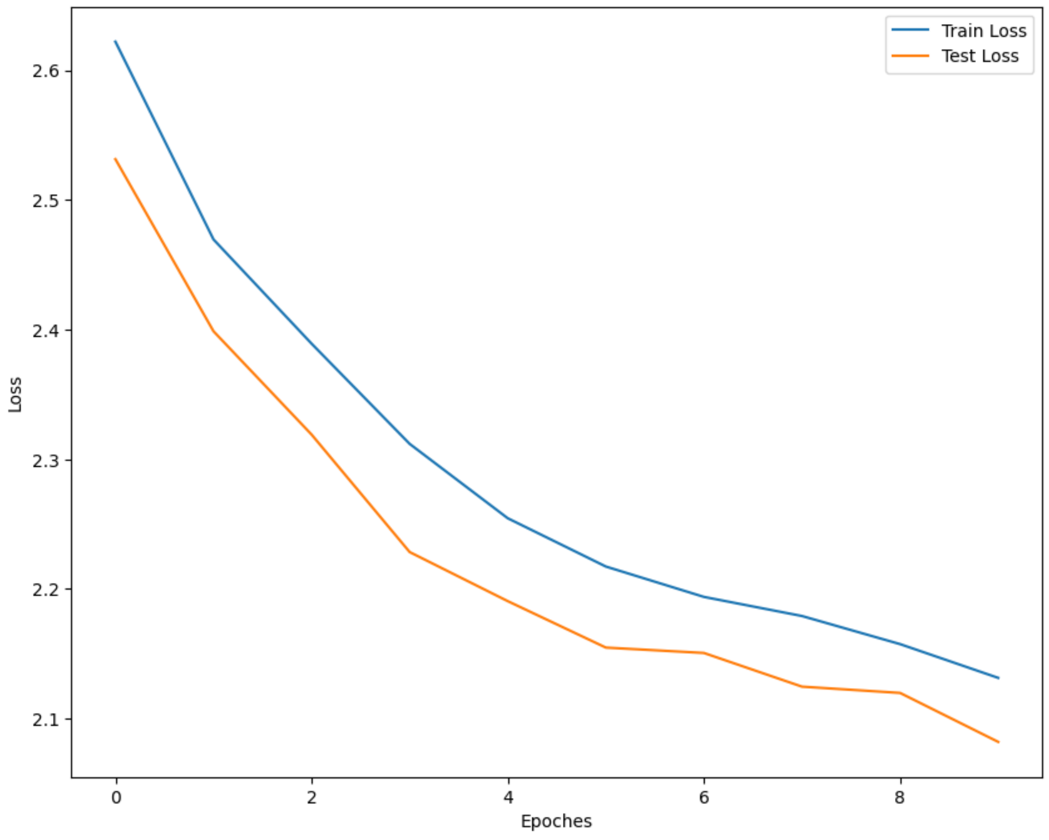
Epoch: 1 / 10
Train Loss: 2.6222 | Test Loss: 2.5315

Epoch: 2 / 10
Train Loss: 2.4696 | Test Loss: 2.3988

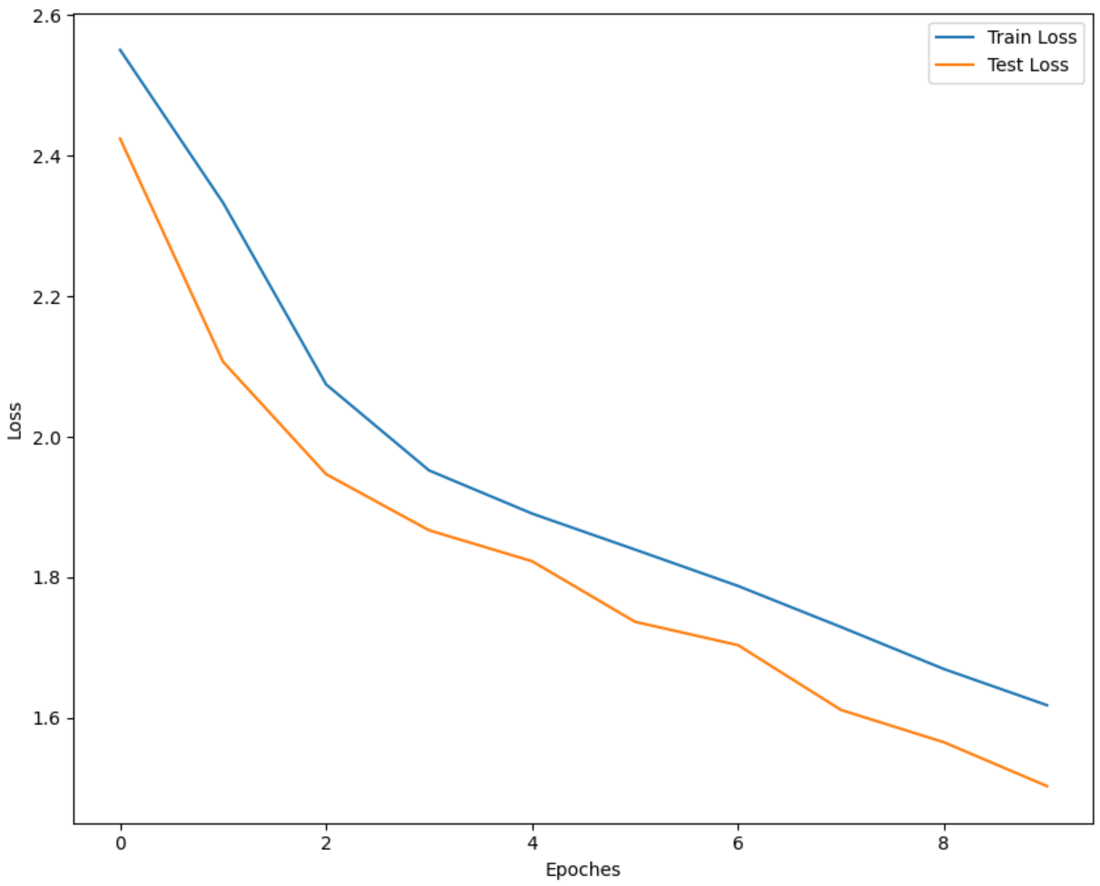
Epoch: 3 / 10
Train Loss: 2.3892 | Test Loss: 2.3191

Epoch: 4 / 10

```
Train Loss: 2.3119 | Test Loss: 2.2285

Epoch: 5 / 10
Train Loss: 2.2546 | Test Loss: 2.1906

Epoch: 6 / 10
Train Loss: 2.2173 | Test Loss: 2.1548

Epoch: 7 / 10
Train Loss: 2.1939 | Test Loss: 2.1507

Epoch: 8 / 10
Train Loss: 2.1792 | Test Loss: 2.1247

Epoch: 9 / 10
Train Loss: 2.1575 | Test Loss: 2.1198

Epoch: 10 / 10
Train Loss: 2.1315 | Test Loss: 2.0821

Total Exe Time: 305.6570204640011 Sec
```



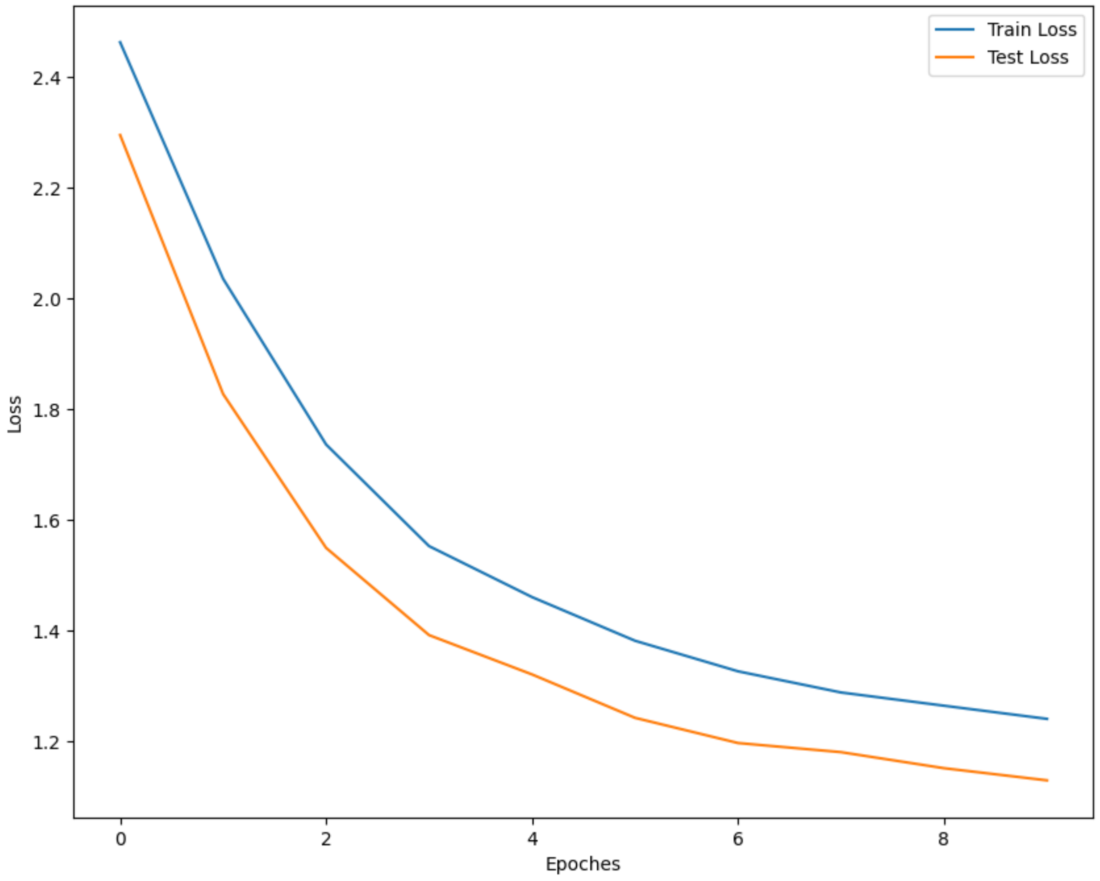
```
Hidden size : 16

Epoch: 1 / 10
Train Loss: 2.5508 | Test Loss: 2.4244

Epoch: 2 / 10
Train Loss: 2.3331 | Test Loss: 2.1067

Epoch: 3 / 10
Train Loss: 2.0746 | Test Loss: 1.9467

Epoch: 4 / 10
Train Loss: 1.9520 | Test Loss: 1.8668

Epoch: 5 / 10
Train Loss: 1.8906 | Test Loss: 1.8227

Epoch: 6 / 10
Train Loss: 1.8393 | Test Loss: 1.7365

Epoch: 7 / 10
Train Loss: 1.7875 | Test Loss: 1.7033

Epoch: 8 / 10
Train Loss: 1.7291 | Test Loss: 1.6110

Epoch: 9 / 10
Train Loss: 1.6692 | Test Loss: 1.5649

Epoch: 10 / 10
Train Loss: 1.6177 | Test Loss: 1.5025

Total Exe Time: 214.62258912900143 Sec
```



Hidden size : 32

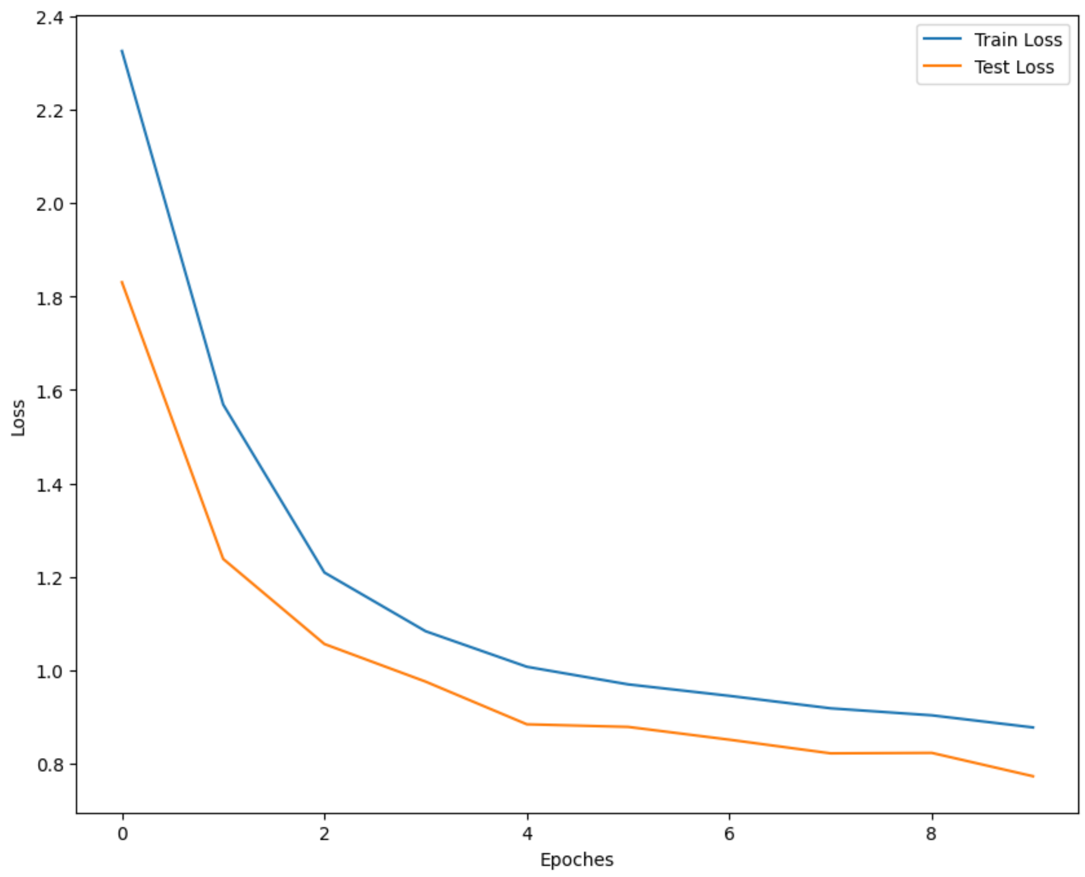
```
Epoch: 1 / 10
Train Loss: 2.4620 | Test Loss: 2.2944

Epoch: 2 / 10
Train Loss: 2.0349 | Test Loss: 1.8268

Epoch: 3 / 10
Train Loss: 1.7357 | Test Loss: 1.5490

Epoch: 4 / 10
Train Loss: 1.5522 | Test Loss: 1.3915

Epoch: 5 / 10
Train Loss: 1.4602 | Test Loss: 1.3206

Epoch: 6 / 10
Train Loss: 1.3814 | Test Loss: 1.2422

Epoch: 7 / 10
```

```
Train Loss: 1.3262 | Test Loss: 1.1967

Epoch: 8 / 10
Train Loss: 1.2880 | Test Loss: 1.1803

Epoch: 9 / 10
Train Loss: 1.2641 | Test Loss: 1.1513

Epoch: 10 / 10
Train Loss: 1.2403 | Test Loss: 1.1293

Total Exe Time: 258.05460119000054 Sec
```



```
Hidden size : 64

Epoch: 1 / 10
Train Loss: 2.3251 | Test Loss: 1.8303

Epoch: 2 / 10
Train Loss: 1.5687 | Test Loss: 1.2383
```

```
Epoch: 3 / 10
Train Loss: 1.2092 | Test Loss: 1.0562

Epoch: 4 / 10
Train Loss: 1.0835 | Test Loss: 0.9757

Epoch: 5 / 10
Train Loss: 1.0075 | Test Loss: 0.8843

Epoch: 6 / 10
Train Loss: 0.9698 | Test Loss: 0.8788

Epoch: 7 / 10
Train Loss: 0.9452 | Test Loss: 0.8515

Epoch: 8 / 10
Train Loss: 0.9186 | Test Loss: 0.8222

Epoch: 9 / 10
Train Loss: 0.9036 | Test Loss: 0.8232

Epoch: 10 / 10
Train Loss: 0.8777 | Test Loss: 0.7732

Total Exe Time: 357.4732475150013 Sec
```



RNN:

Hidden size : 8

Epoch: 1 / 10
Train Loss: 2.8980 | Test Loss: 2.7080

Epoch: 2 / 10
Train Loss: 2.6882 | Test Loss: 2.6406

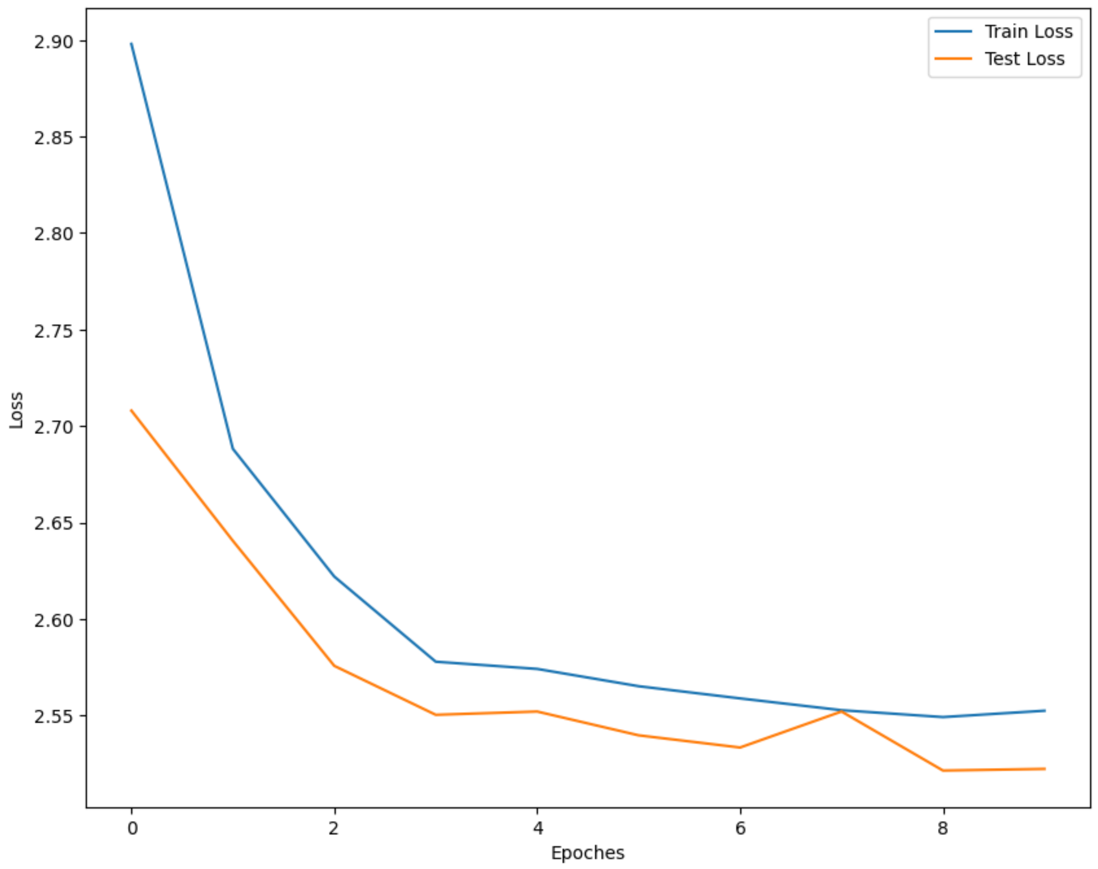
Epoch: 3 / 10
Train Loss: 2.6220 | Test Loss: 2.5758

Epoch: 4 / 10
Train Loss: 2.5779 | Test Loss: 2.5504

Epoch: 5 / 10
Train Loss: 2.5742 | Test Loss: 2.5521

Epoch: 6 / 10
Train Loss: 2.5652 | Test Loss: 2.5398

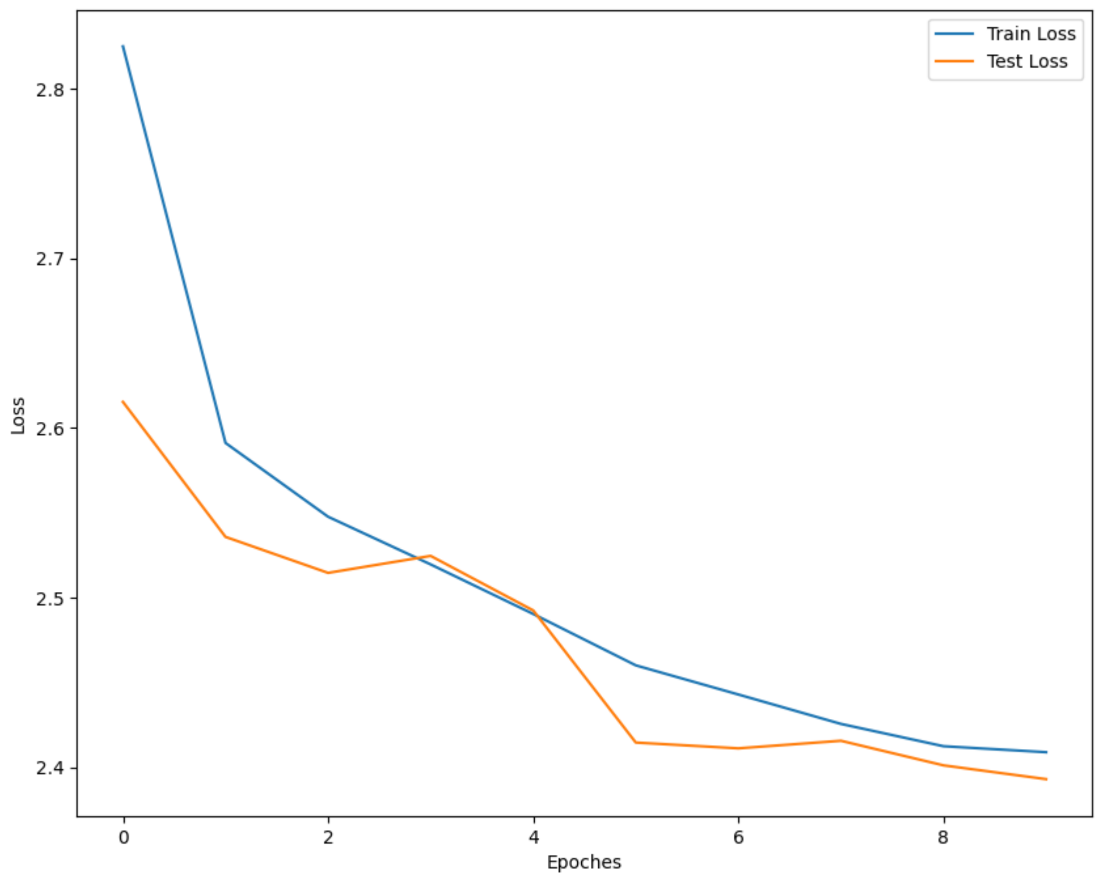
```
Epoch: 7 / 10
Train Loss: 2.5589 | Test Loss: 2.5334

Epoch: 8 / 10
Train Loss: 2.5528 | Test Loss: 2.5521

Epoch: 9 / 10
Train Loss: 2.5492 | Test Loss: 2.5215

Epoch: 10 / 10
Train Loss: 2.5525 | Test Loss: 2.5224

Total Exe Time: 160.84545144000003 Sec
```



Hidden size : 16

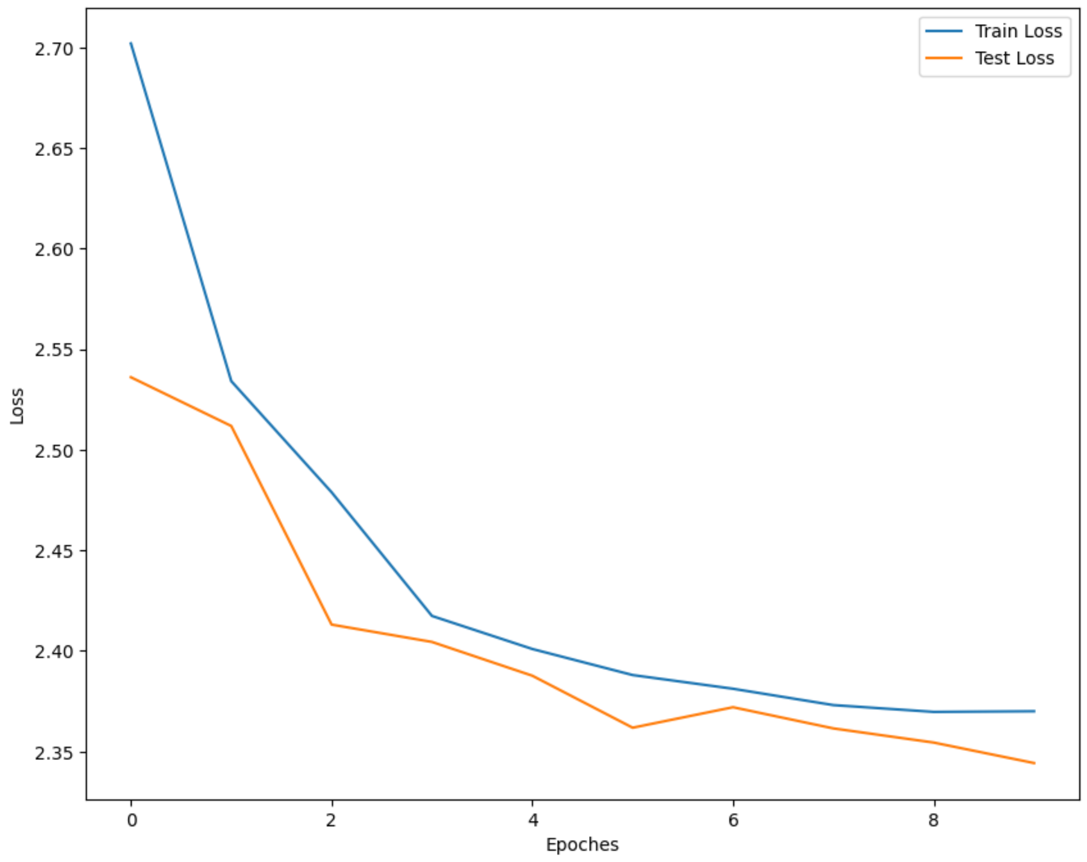
```
Epoch: 1 / 10
Train Loss: 2.8248 | Test Loss: 2.6153

Epoch: 2 / 10
Train Loss: 2.5911 | Test Loss: 2.5357
```

```
Epoch: 3 / 10
Train Loss: 2.5476 | Test Loss: 2.5145

Epoch: 4 / 10
Train Loss: 2.5195 | Test Loss: 2.5245

Epoch: 5 / 10
Train Loss: 2.4903 | Test Loss: 2.4924

Epoch: 6 / 10
Train Loss: 2.4600 | Test Loss: 2.4145

Epoch: 7 / 10
Train Loss: 2.4428 | Test Loss: 2.4111

Epoch: 8 / 10
Train Loss: 2.4255 | Test Loss: 2.4155

Epoch: 9 / 10
Train Loss: 2.4123 | Test Loss: 2.4010

Epoch: 10 / 10
Train Loss: 2.4088 | Test Loss: 2.3929

Total Exe Time: 156.07999123099944 Sec
```



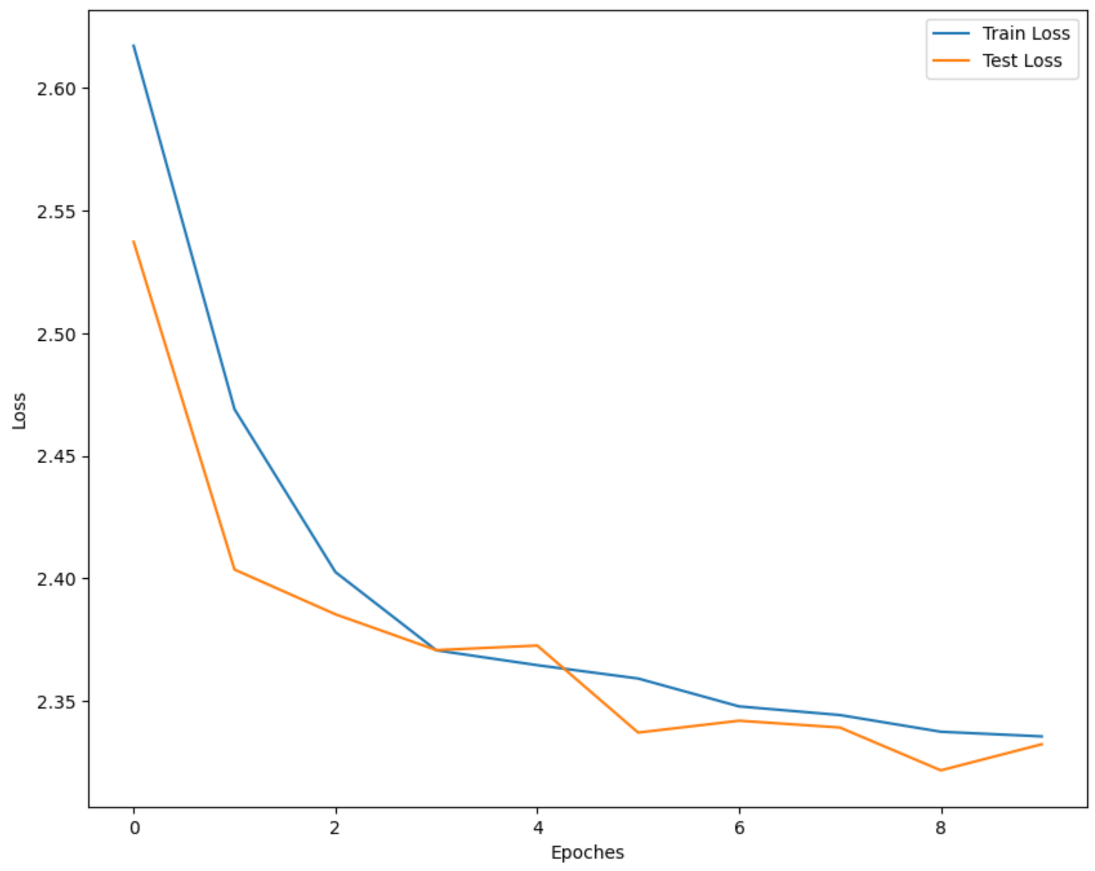
```
Hidden size : 32
Epoch: 1 / 10
Train Loss: 2.7020 | Test Loss: 2.5361

Epoch: 2 / 10
Train Loss: 2.5341 | Test Loss: 2.5118

Epoch: 3 / 10
Train Loss: 2.4789 | Test Loss: 2.4131

Epoch: 4 / 10
Train Loss: 2.4174 | Test Loss: 2.4045

Epoch: 5 / 10
Train Loss: 2.4010 | Test Loss: 2.3877

Epoch: 6 / 10
Train Loss: 2.3880 | Test Loss: 2.3618

Epoch: 7 / 10
Train Loss: 2.3812 | Test Loss: 2.3720
```

```
Epoch: 8 / 10
Train Loss: 2.3731 | Test Loss: 2.3615

Epoch: 9 / 10
Train Loss: 2.3697 | Test Loss: 2.3545

Epoch: 10 / 10
Train Loss: 2.3700 | Test Loss: 2.3443

Total Exe Time: 163.64969378000023 Sec
```



```
Epoch: 1 / 10
Train Loss: 2.6173 | Test Loss: 2.5373

Epoch: 2 / 10
Train Loss: 2.4689 | Test Loss: 2.4035

Epoch: 3 / 10
Train Loss: 2.4025 | Test Loss: 2.3853
```

```
Epoch: 4 / 10
Train Loss: 2.3706 | Test Loss: 2.3707

Epoch: 5 / 10
Train Loss: 2.3645 | Test Loss: 2.3725

Epoch: 6 / 10
Train Loss: 2.3591 | Test Loss: 2.3370

Epoch: 7 / 10
Train Loss: 2.3477 | Test Loss: 2.3418

Epoch: 8 / 10
Train Loss: 2.3442 | Test Loss: 2.3391

Epoch: 9 / 10
Train Loss: 2.3373 | Test Loss: 2.3216

Epoch: 10 / 10
Train Loss: 2.3354 | Test Loss: 2.3322

Total Exe Time: 175.14109500199993 Sec
```

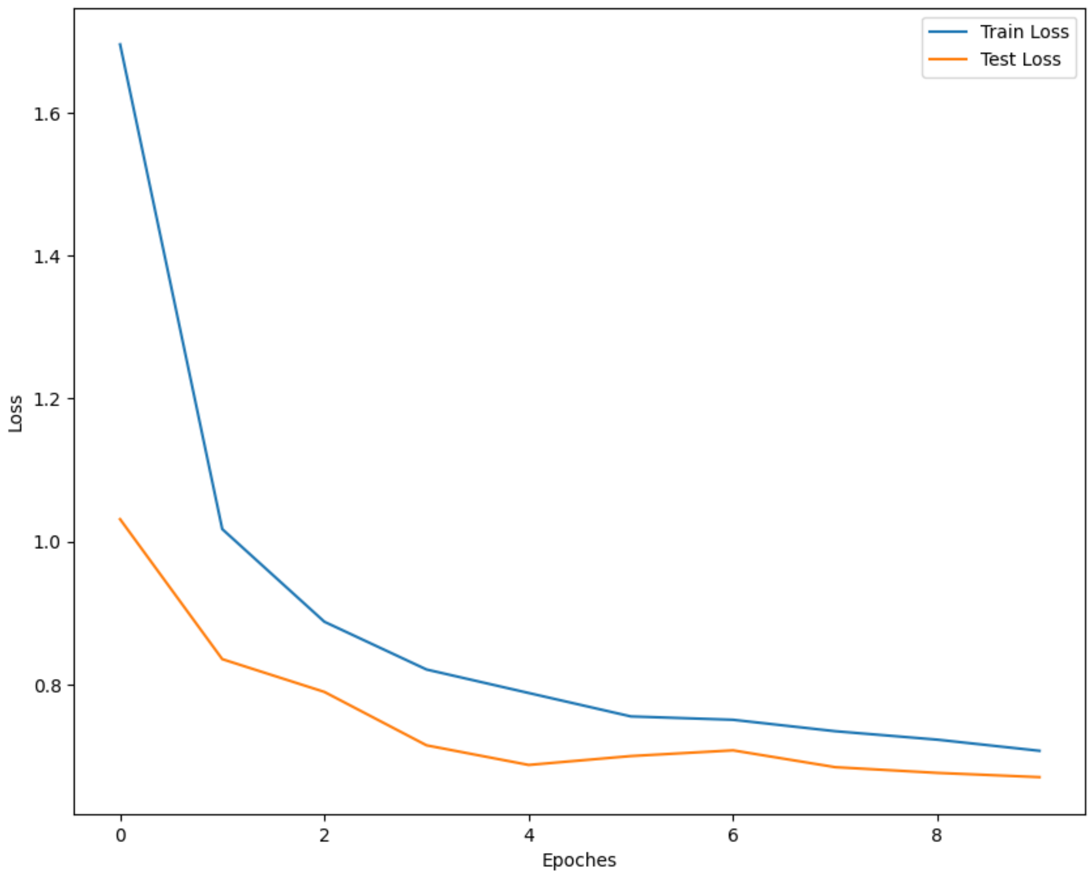


Observation:

- For Both LSTM and RNN, more hidden nodes perform better than less number of hidden nodes.
- However, when the number of hidden nodes increases execution time also increases.
- RNN- and LSTM-based seq2seq models' performance can be significantly impacted by the size of the hidden layer. With larger hidden layer sizes, the model can capture more intricate patterns in the data, whereas with smaller hidden layer sizes, the model may be less capable of modeling and may be less able to catch fine-grained characteristics. When the size of the hidden layer is too small, the model could have trouble capturing the nuances of the input data, which could result in poorer accuracy or subpar performance. But bigger hidden layer sizes also call for greater computational power and might be more prone to overfitting, particularly if the dataset is tiny.

D

```
Epoch: 1 / 10
Train Loss: 1.6954 | Test Loss: 1.0312

Epoch: 2 / 10
Train Loss: 1.0174 | Test Loss: 0.8354

Epoch: 3 / 10
Train Loss: 0.8877 | Test Loss: 0.7896

Epoch: 4 / 10
Train Loss: 0.8209 | Test Loss: 0.7149

Epoch: 5 / 10
Train Loss: 0.7881 | Test Loss: 0.6874

Epoch: 6 / 10
Train Loss: 0.7553 | Test Loss: 0.6999

Epoch: 7 / 10
Train Loss: 0.7506 | Test Loss: 0.7079

Epoch: 8 / 10
Train Loss: 0.7346 | Test Loss: 0.6843

Epoch: 9 / 10
Train Loss: 0.7228 | Test Loss: 0.6763

Epoch: 10 / 10
Train Loss: 0.7072 | Test Loss: 0.6704

Total Exe Time: 654.0894354630001 Sec
```



I already train without attention to LSTM in Part A so use Part A result for comparison.

Observation:

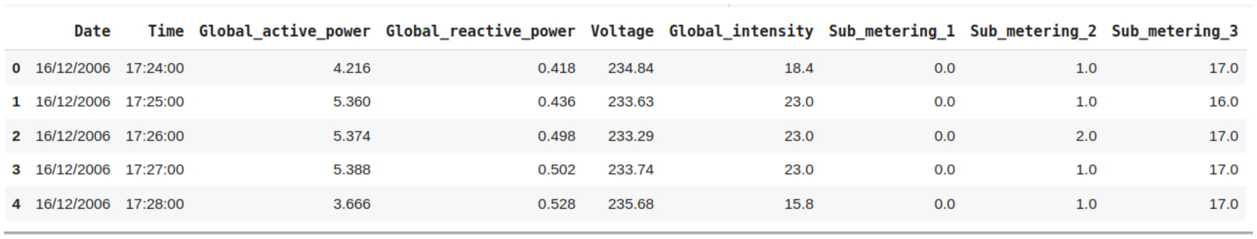
- Without attentionLSTM S2S loss decreased very well from 2.26 to 0.90 (train) and (test) from 1.75 to 0.83.
- With attentionLSTM S2S loss decreased very well from 1.69 to 0.70 (train) and (test) from 1.03 to 0.67.
- However due to the attention layer Attention LSTM takes more time than without attention LSTM.
- Here clearly shows that loss reduction is also more in attention LSTM. So, attention LSTM Seq2Seq performs better than without attention LSTM Seq2Seq.

Question 2

- Import require libraries
- Write device agnostic code
- Load dataset
- Preprocess on it
- Split it in 80:20
- Now make dataloader of them
- Write train and test loop
- Function for loss vs epochs
- Build LSTM model for prediction
- Now train model for 80:20 data
- After 80:20 model train split data in to 70;30 and again train model

Result:

Raw data

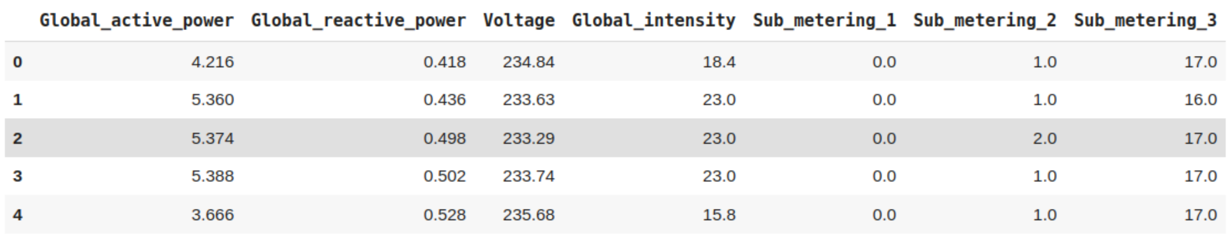


	Date	Time	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
0	16/12/2006	17:24:00	4.216	0.418	234.84	18.4	0.0	1.0	17.0
1	16/12/2006	17:25:00	5.360	0.436	233.63	23.0	0.0	1.0	16.0
2	16/12/2006	17:26:00	5.374	0.498	233.29	23.0	0.0	2.0	17.0
3	16/12/2006	17:27:00	5.388	0.502	233.74	23.0	0.0	1.0	17.0
4	16/12/2006	17:28:00	3.666	0.528	235.68	15.8	0.0	1.0	17.0

Nan value in data:

```
Null value in Global_active_power: 25979
Null value in Global_reactive_power: 25979
Null value in Voltage: 25979
Null value in Global_intensity: 25979
Null value in Sub_metering_1: 25979
Null value in Sub_metering_2: 25979
Null value in Sub_metering_3: 25979
```

Preprocess data



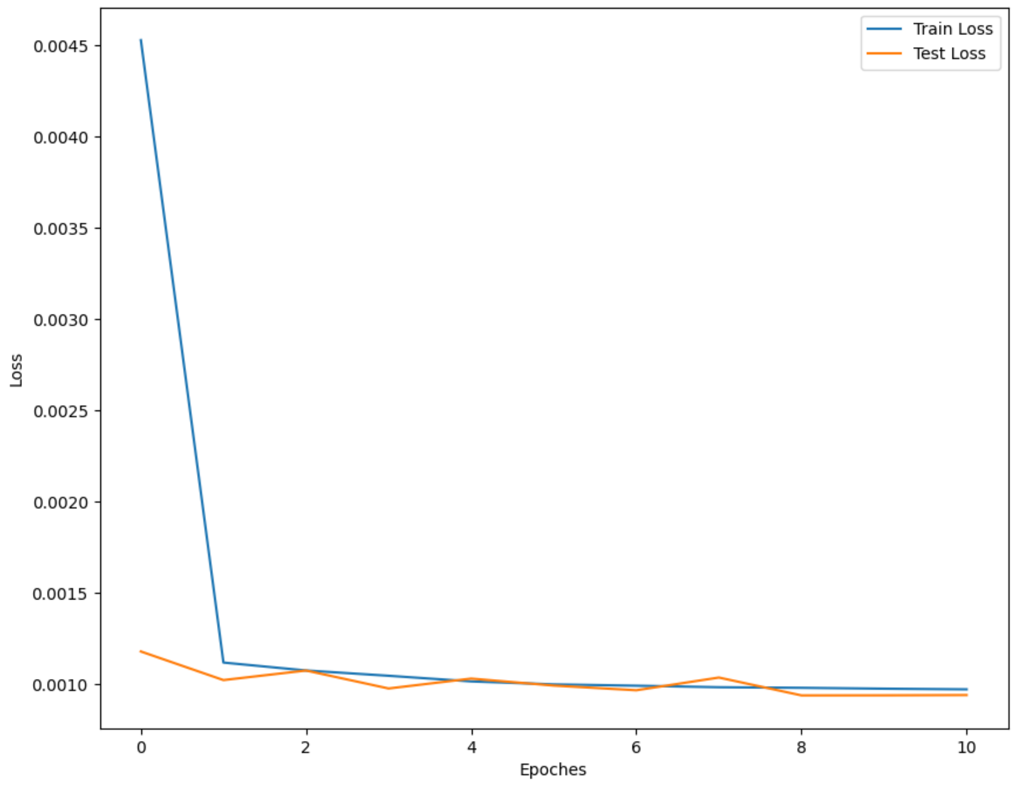
	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3
0	4.216	0.418	234.84	18.4	0.0	1.0	17.0
1	5.360	0.436	233.63	23.0	0.0	1.0	16.0
2	5.374	0.498	233.29	23.0	0.0	2.0	17.0
3	5.388	0.502	233.74	23.0	0.0	1.0	17.0
4	3.666	0.528	235.68	15.8	0.0	1.0	17.0

Nan value in data after preprocess:

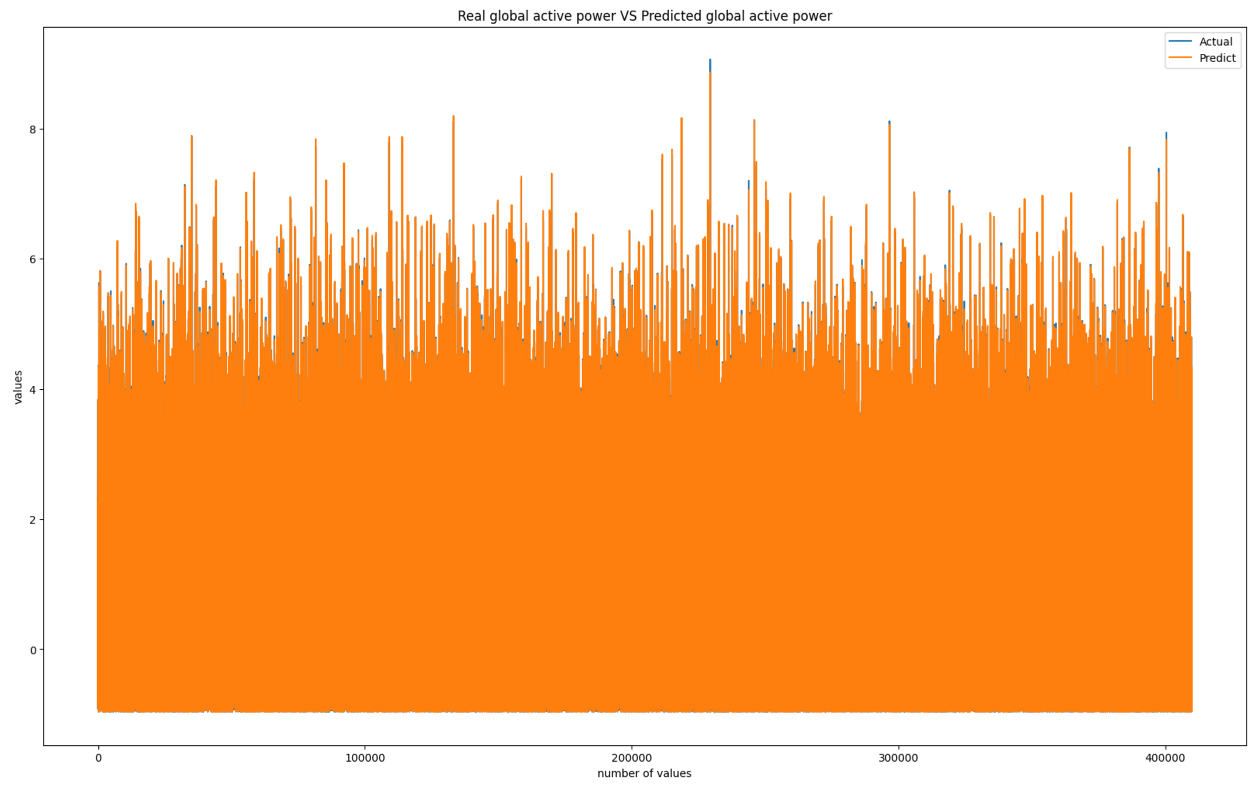
```
Null value in Global_active_power: 0
Null value in Global_reactive_power: 0
Null value in Voltage: 0
Null value in Global_intensity: 0
Null value in Sub_metering_1: 0
Null value in Sub_metering_2: 0
Null value in Sub_metering_3: 0
```

80:20 result:

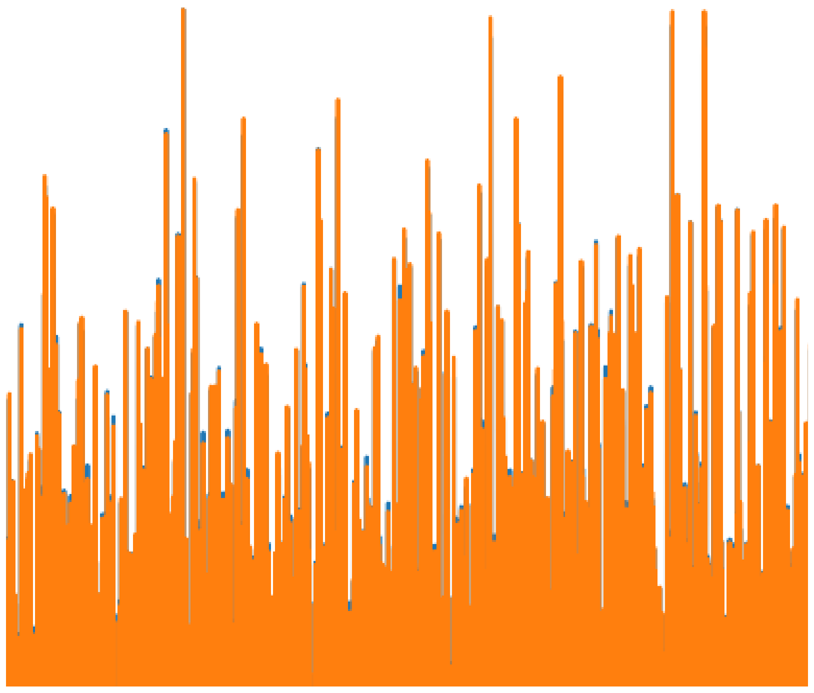
Loss: run time 11/11 [15:51<00:00, 88.93s/it]
Epoch: 1 Train Loss: 0.0045 | Test Loss: 0.0012
Epoch: 2 Train Loss: 0.0011 | Test Loss: 0.0010
Epoch: 3 Train Loss: 0.0011 | Test Loss: 0.0011
Epoch: 4 Train Loss: 0.0010 | Test Loss: 0.0010
Epoch: 5 Train Loss: 0.0010 | Test Loss: 0.0010
Epoch: 6 Train Loss: 0.0010 | Test Loss: 0.0010
Epoch: 7 Train Loss: 0.0010 | Test Loss: 0.0010
Epoch: 8 Train Loss: 0.0010 | Test Loss: 0.0010
Epoch: 9 Train Loss: 0.0010 | Test Loss: 0.0009
Epoch: 10 Train Loss: 0.0010 | Test Loss: 0.0009
Epoch: 11 Train Loss: 0.0010 | Test Loss: 0.0009



Actual data vs Predicted data curve



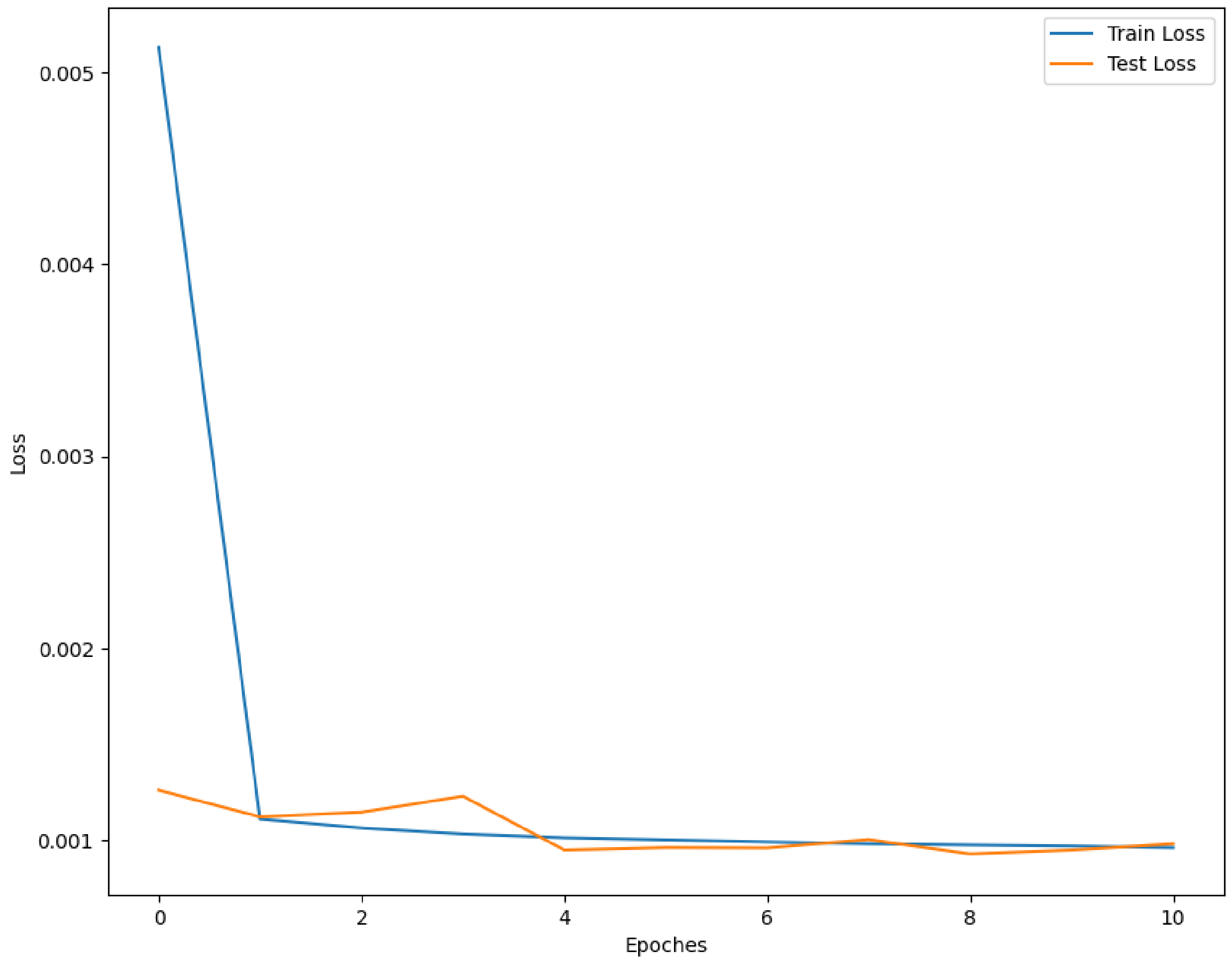
Zoom image for bette view



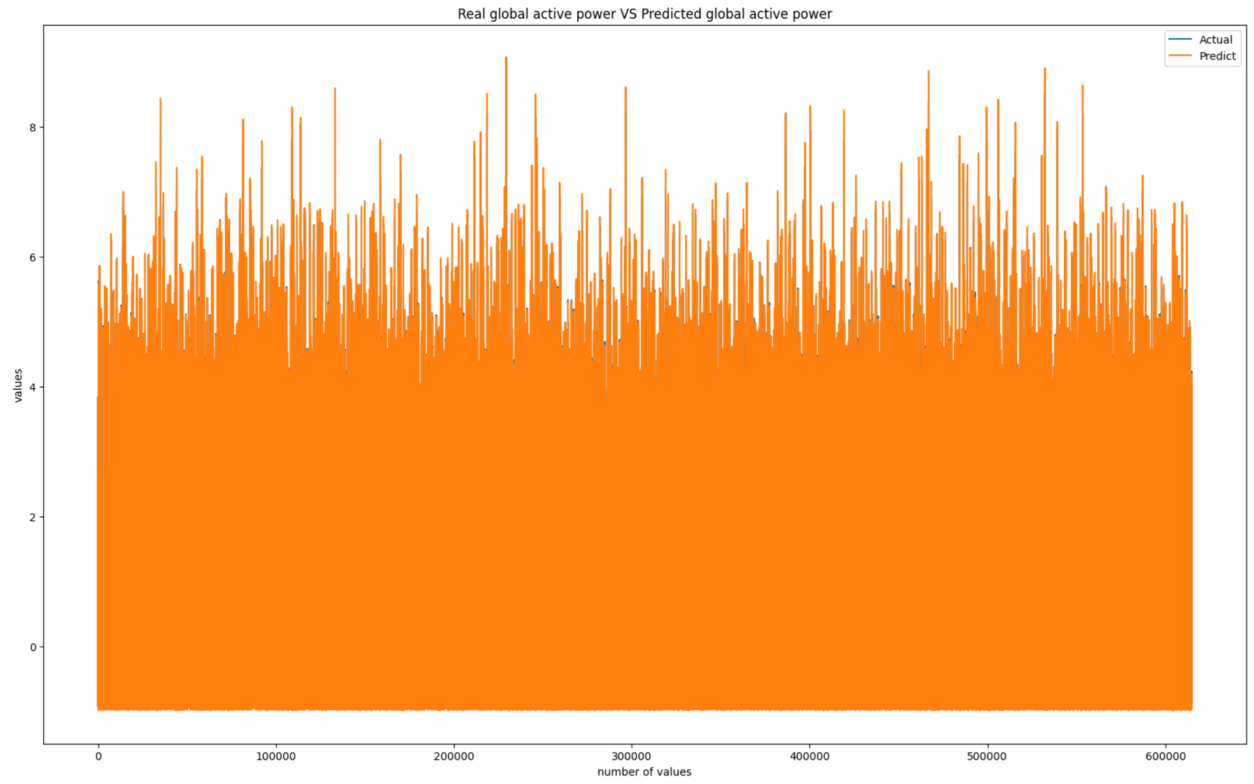
70:30 result:

Loss: run time: 11/11 [13:00<00:00, 70.52s/it]

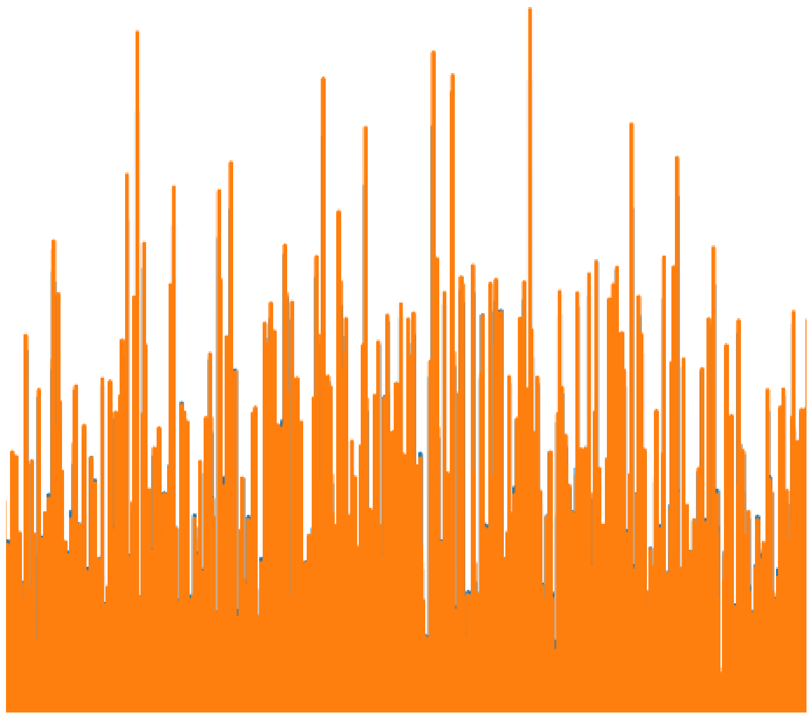
```
Epoch: 1  Train Loss: 0.0051 | Test Loss: 0.0013
Epoch: 2  Train Loss: 0.0011 | Test Loss: 0.0011
Epoch: 3  Train Loss: 0.0011 | Test Loss: 0.0011
Epoch: 4  Train Loss: 0.0010 | Test Loss: 0.0012
Epoch: 5  Train Loss: 0.0010 | Test Loss: 0.0010
Epoch: 6  Train Loss: 0.0010 | Test Loss: 0.0010
Epoch: 7  Train Loss: 0.0010 | Test Loss: 0.0010
Epoch: 8  Train Loss: 0.0010 | Test Loss: 0.0010
Epoch: 9  Train Loss: 0.0010 | Test Loss: 0.0009
Epoch: 10  Train Loss: 0.0010 | Test Loss: 0.0010
Epoch: 11  Train Loss: 0.0010 | Test Loss: 0.0010
```



Actual data vs Predict data



Zoom image for better view:



Observation:

Feature	**80:20 data Split**	**70:30 data Split**
Loss	at most similar	at most similar
Model Train Time	Take more time for same number of epoch due to more train sample	Take less time for same number of epoch due to less train sample
Time taken for 11 epoche	15:51 min	13:00 min
Actual vs predict result	Different of 0.1-0.3 (clearly show in zoom out image)	Nearly same(clearly show in zoom out image)