**Assignment 3: Timeseries:**

Steven Kalinoff  
4/07/2024

Adv. Machine Learning 64061

**PROMPT:**

***Use any or all of the methods we discussed in class to improve weather time-series forecasting problems discussed in class. These methods can include:***

***1. Adjusting the number of units in each recurrent layer in the stacked setup***

***2. Using layer\_lstm() instead of layer\_gru().***

***3. Using a combination of 1d\_convnets and RNN.***

Baseline Data yields a result of MAE 2.44 with the lowest error rate at epoch 4 suggesting the data is starting to overfit after that point so we set that as the target for improvement.

our first approach is to use the LSTM model in the first layer to optimize both runtime and training/validation accuracy. At this point we can see that our first epoch has a horrible time training the first epoch but due to the Long Short Term Memory (LSTM) we get much better margins during the 2nd epoch. We get a large spike in the 5th epoch on the validation size of things pointing to overfitting of the data. We need to address this phenomenon in future iterations of our design.

A graph with blue dots

Description automatically generated

Our next model is based on starting to incorporate bi-directional RNN’s to exploit the time series data. I then run some initial tests to compare the first few epochs with the standard dropout and units in each layer to gage relative accuracy.

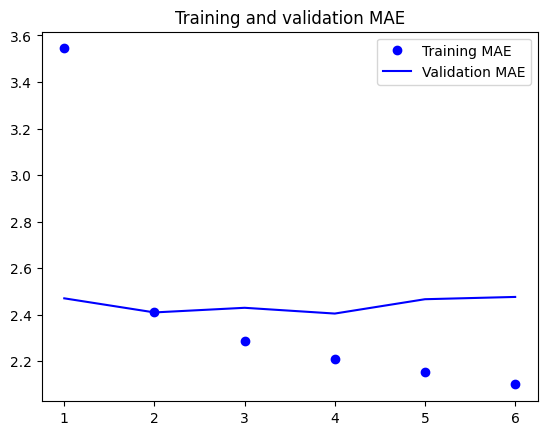
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epoch-test | Training Loss | Training MAE | Validation Loss | Validation MAE |
| 1 | 25.4645 | 3.7208 | 9.3587 | 2.3598 |
| 2 | 14.0821 | 2.9091 | 8.9145 | 2.3215 |
| 3 | 13.2144 | 2.8202 | 9.327 | 2.389 |
| Epoch  Val |  |  |  |  |
| 1 | 26.4448 | 3.6716 | 10.8194 | 2.5541 |
| 2 | 9.4285 | 2.3946 | 10.1216 | 2.468 |
| 3 | 8.5096 | 2.269 | 10.1454 | 2.4719 |
|  |  |  |  |  |
| Test MAE |  |  |  | 2.48 |

A graph with blue dots

Description automatically generated

From the data and comments about the arbitrary nature of selections I keep reiterating this process of adjusting these parameters to find a best outcome and then run it for the full 6 epochs that mirror my initial hypothesis.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epoch-test | Training Loss | Training MAE | Validation Loss | Validation MAE |
| 1 | 34.3266 | 4.281 | 11.0047 | 2.4936 |
| 2 | 13.4494 | 2.8425 | 8.9962 | 2.3048 |
| 3 | 12.5011 | 2.746 | 8.8024 | 2.2845 |
| 4 | 12.0632 | 2.7006 | 8.5991 | 2.2686 |
| 5 | 11.7217 | 2.6634 | 8.4874 | 2.2537 |
| 6 | 11.4808 | 2.6353 | 8.9963 | 2.3268 |
| MAE |  |  |  | 2.46 |
| Epoch-Val |  |  |  |  |
| 1 | 24.2576 | 3.5441 | 10.1482 | 2.4705 |
| 2 | 9.5676 | 2.412 | 9.6563 | 2.4099 |
| 3 | 8.596 | 2.2852 | 9.8055 | 2.4298 |
| 4 | 8.0521 | 2.2114 | 9.6615 | 2.405 |
| 5 | 7.6213 | 2.1528 | 10.2233 | 2.4667 |
| 6 | 7.2401 | 2.1003 | 10.3242 | 2.4765 |



Our Training data is improving but the validation portion is flatlining if not increasing so we may need to rethink our approach. In the last model both dropout and recurrent dropout methods are incorporated simultaneously to combat overfitting in and in-between the neurons. This method should be more selective and by dropping the rate to .25 for both we should let more data pass through for learning without the risk of overfitting.

We then implement a checkpoint to save the best model performance and details then loading that model the next running epoch.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Type** | **Epoch** | **Train Loss** | **Train MAE** | **Val Loss** | **Val MAE** |
| LSTM | 1 | 38.9456 | 4.5507 | 12.3292 | 2.6458 |
| LSTM | 2 | 13.5866 | 2.852 | 9.2745 | 2.3654 |
| LSTM | 3 | 12.4571 | 2.739 | 9.0868 | 2.3264 |
| LSTM | 4 | 11.9522 | 2.6851 | 9.3305 | 2.3569 |
| LSTM | 5 | 11.5 | 2.6326 | 8.8503 | 2.2965 |
| LSTM | 6 | 11.1972 | 2.596 | 8.9069 | 2.3067 |
| GRU | 1 | 26.8214 | 3.7189 | 10.2932 | 2.4829 |
| GRU | 2 | 9.5152 | 2.4046 | 10.1077 | 2.461 |
| GRU | 3 | 8.467 | 2.2604 | 10.3806 | 2.4808 |
| GRU | 4 | 7.9102 | 2.1827 | 10.1626 | 2.4438 |
| GRU | 5 | 7.5275 | 2.1288 | 10.8214 | 2.5122 |
| GRU | 6 | 7.2864 | 2.0928 | 10.1888 | 2.4541 |

A graph with red and blue lines

Description automatically generated

On the final model we can see side-by-side the GRU vs LSTM models with identical parameters and training/validation data. They both rapidly converge after the 0th epoch and GRU model seems to be offset in its ability to validate training data. At this point both models do not appear to be overfitting and with the available data will likely see incremental progress over further training epochs. Overall the GRU seems to be best optimized for speed with each epoch taking about 100s less to train but the LSTM model yields better validation data. Lastly with the inversion of training loss and validation loss on the 2 models we appear to be close to a well fitting model as higher levels of validation loss over training loss usually points to a model be underfit and vis versa usually points to overfitting.

Running test data:

With the test data being ran in the last configuration we see the GRU model has less learning loss and a lower MAE and this model also runs faster than the LSTM method suggesting it is the superior method for evaluating the test data.

GRU Test Loss: 9.9

GRU Test MAE: 2.4

LSTM Test Loss: 11.3

LSTM Test MAE: 2.6