CSC 578 Final Project Part A TIME SERIES FORECASTING
Sojeong Yang

Kaggle username: Soji, Ranking: Public-29 (time: 5:43PM)

Brief Explanation of Baseline Model

Model Architecture:

```
base_model = tf.keras.models.Sequential([
    # Shape [batch, time, features] => [batch, time, lstm_units]
    tf.keras.layers.LSTM(32, return_sequences=True),
    # Shape => [batch, time, features]
    tf.keras.layers.Dense(units=1)
])
```

- batch_size: 32
- number of recurrent units: 32
- recurrent layers: 1
- optimizer: tf.optimizers.Adam()
- MAX_EPOCHS = 50
- recurrent_dropout: N/A
- return_sequences: True

A baseline is good to have when comparing to the more complicated model. Our task is to predict future traffic volume at a location. LSTM is a type of Recurrent Neural Networks and is commonly applied when order is an important feature element. The important feature of LSTM is that, as mentioned above, it stores sequences and uses them for learning. Therefore, I picked the LSTM model as the base model for this project of time series forecasting. Also, LSTM is a model that improves the long-term dependency and the gradient vanishing problems that occur during backpropagation.

I used a LSTM layer with 32 internal units. I planned to experiment with stacking RNN layers for a more complicated model, so I used only one layer to make the base model simple. Also, I used a default batch size, 32 for this base model. To minimize the loss value, I planned to experiment with different batch sizes, for example, batch_size = 16, 32, 64, 128, 256. 'return_sequences' is related to the output of an LSTM layer. When 'return_sequences' set to 'False', the shape of the output is (batch_size, LSTM units). On the other hand, 'return_sequences' set to 'True', the shape of the output is (batch_size, timesteps, LSTM units). To be more specific, a single LSTM layer contains a LSTM cell and each LSTM cell is going to output one hidden state for each input. If we set the 'return_sequences = True', we gain access to the hidden state output. For the base model that has a single layer, I set 'return_sequences = True' because I am interested in the all hidden state outputs corresponding to each timestep that holds information about each input sequence.

Result of the base model:

```
302/302 [==============================] - 0s 1ms/step - loss: 0.1947 - mean_absolute_error: 0.3024
```

CSC 578 Final Project Part A TIME SERIES FORECASTING
Sojeong Yang

Kaggle username: Soji, Ranking: Public-29 (time: 5:43PM)

Brief Explanation of the Best Model

Model Architecture:

```
lstm_model = tf.keras.models.Sequential([

    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32, return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences=True)),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences=True)),

    tf.keras.layers.Dense(units=1)
])
```

- Bidirectional RNNs
- batch_size: 16
- number of recurrent units: 32, 64, 64
- recurrent layers: 3
- optimizer: tf.optimizers.Adam()
- MAX_EPOCHS = 50
- recurrent_dropout: N/A
- return_sequences: True

Bidirectional RNNs are literally bidirectional RNN layers. This is characterized by not only considering the previous information, but also the latter information. There is a hidden layer that has the previous state information and the hidden layer that has the backward state information. Inverted input values are put in the forward hidden layer and the rear hidden layer. And the value of the output layer is learned by applying it after all input values have been calculated to the hidden layers in both directions. Also, similar to stacked LSTM layers, we should set the 'return_sequences' to 'True' because it is required that a LSTM layer, previous to each subsequent LSTM layer, must return the sequence. When I tried bidirectional LSTM model after the stacked LSTM, it showed a lower value of mean absolute error. So, I continued to try bidirectional LSTMs with different number of units, different numbers of stacked bidirectional layers, added recurrent dropout, stateful=True and so on. However, the simple 32-64-64 bidirectional LSTM model gave me the lowest value of mean absolute error.

Result of the base model:

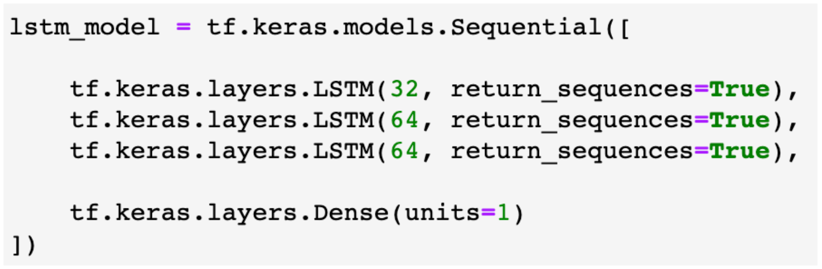
```
603/603 [==============================] - 2s 3ms/step - loss: 0.0972 - mean_absolute_error:
0.2066
```

CSC 578 Final Project Part A TIME SERIES FORECASTING
Sojeong Yang

Kaggle username: Soji, Ranking: Public-29 (time: 5:43PM)

Brief Explanation of Additional Model 2

Model Architecture:

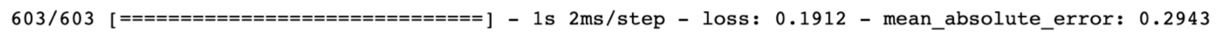


```
lstm_model = tf.keras.models.Sequential([

    tf.keras.layers.LSTM(32, return_sequences=True),
    tf.keras.layers.LSTM(64, return_sequences=True),
    tf.keras.layers.LSTM(64, return_sequences=True),

    tf.keras.layers.Dense(units=1)
])
```

- Stacking recurrent layers
- batch_size: 16
- number of recurrent units: 32, 64, 64
- recurrent layers: 3
- optimizer: tf.optimizers.Adam()
- MAX_EPOCHS = 50
- recurrent_dropout: N/A
- return_sequences: True

This model is stacked LSTM layers. It made a deep network architecture by stacking RNN layers. It is important thing to make the stacked LSTM layers, so we should set the 'return_sequences' to 'True' because it is required that a LSTM layer previous to each subsequent LSTM layer must return the sequence. I tried 2 stacked LSTMs with a different number of recurrent units and 3 stacked LSTMs and 4 stacked LSTMs. According to experiment results, 3 stacked LSTMs and '32+64+64' number of recurrent units gave the minimum value of mean absolute error.

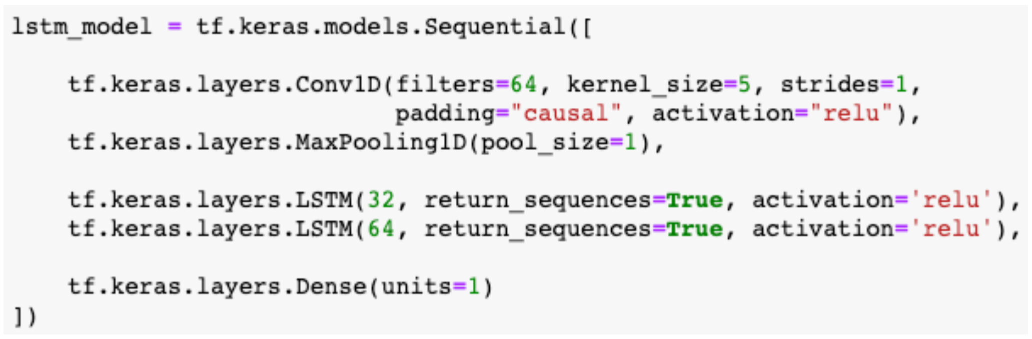
Result of the model 2:



```
603/603 [==============================] - 1s 2ms/step - loss: 0.1912 - mean_absolute_error: 0.2943
```

Brief Explanation of Additional Model 3

Model Architecture:



```
lstm_model = tf.keras.models.Sequential([

    tf.keras.layers.Conv1D(filters=64, kernel_size=5, strides=1,
                           padding="causal", activation="relu"),
    tf.keras.layers.MaxPooling1D(pool_size=1),

    tf.keras.layers.LSTM(32, return_sequences=True, activation='relu'),
    tf.keras.layers.LSTM(64, return_sequences=True, activation='relu'),

    tf.keras.layers.Dense(units=1)
])
```

CSC 578 Final Project Part A TIME SERIES FORECASTING
Sojeong Yang

Kaggle username: Soji, Ranking: Public-29 (time: 5:43PM)

- 1D convnet + RNN
- batch_size: 16
- number of recurrent units: 32, 64
- recurrent layers: 2
- optimizer: tf.optimizers.Adam()
- MAX_EPOCHS = 50
- recurrent_dropout: 0.1
- return_sequences: True

This model is a hybrid model which combines CNN and LSTM. 2D Convolutional Neural Network models that we leaned were originated for image classification. In this case, the model receives a 2-dimensional input representing an image's pixels and color channels, in a feature learning process. Likewise, this similar process can be applied to 1-dimensional sequences of data. The model takes out features from sequences data and maps the internal features of the sequence. '1D CNN' is very effective for extract features from a fixed-length segment of the overall dataset and the location is not important relative to where the feature is located in the segment.

Result of the model 2:

```
603/603 [==============================] - 1s 2ms/step - loss: 0.2011 - mean_absolute_error:
0.3065
```

Reflection on your result and the competition

The experiments that changed the hyperparameters were very interesting. Time flew while I was working on it. I was so curious and could not wait to receive the results whenever I changed the hyperparameters. However, there are so many different combinations that I can use for hyperparameters which makes this complicated. I also struggled to make 'stateful RNN' work. I thought that I provided the right 'batch_input_shape' in the first LSTM layer. I checked that I changed 'shuffle=False' in 'model.fit()' because it is one of the conditions to make 'stateful RNN'. While I did everything that I need to do, I still got errors. Finally, I set the right 'batch_input_shape' and I was able to run without error which made me feel very successful. The competition made me competitive about my result vs. other classmates' results. So, I tried all kinds of advanced features and architectures from the instructions, but I only minimized the value of MAE by about 0.1 point. I want to figure it out more and improve my model even though the competition is finished. I found that I enjoyed this project so much.

Reflection on the project as well as the course overall

I took most of time to understand and work through the 'tensorflow time series tutorial'. I was confused about the input shape and output shape and how that would meet our goal of predicting a 6-hour input window and 3 hours past the end of that window. So, I watched our lecture and studied the lecture slides more carefully. After I created a submission file, I felt great that I got results. This course and assignments were challenging to me and every week, I took lots of time to understand the concepts and get good results for assignments. As I look back on the course, I feel the course was incredibly satisfying and that I improved a lot. I believe the knowledge that I received from this course will be beneficial to my future and look forward to continuing to learn more on the subject.

Reference

- Time series forecasting, TensorFlow, website: https://www.tensorflow.org/tutorials/structured_data/time_series#recurrent_neural_network

- Multivariate Time Series Forecasting with LSTMs in Keras, Machine Learning Mastery, website: https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/

- Working with RNNs, website: https://keras.io/guides/working_with_rnns/

- tf.keras.layers.GRU, TensorFlow, Website: https://www.tensorflow.org/api_docs/python/tf/keras/layers/GRU

- How to Develop a Bidirectional LSTM For Sequence Classification in Python with Keras, Machine Learning Mastery, website: https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/