National Tsing Hua University

11320IEEM 513600

Deep Learning and Industrial Applications

Homework 4

Name: 李宜庭 Student ID: 112030751

Due on 2025/05/01.

Note: DO NOT exceed 3 pages.

1. (15 points) Experiment with different window sizes and steps. Train the model using **3** different combinations of window size and step. Evaluate the Mean Squared Error (MSE) for each configuration. Report the MSEs using a table and analyze the results. (Approximately 100 words.)

Model Setting: hidden_dim=500, num_layers=1, lr=0.001, epochs=100, unnormalized training data, features=(Open, High, Low, Close)

Window Size	Step Size	Test MSE (Raw Input)
10	5	17.67
10	10	159.79
5	10	161.61

From the above result, the configuration (10, 5) achieved the lowest test MSE of 17.67, while (10, 10) and (5, 10) performed significantly worse, with MSEs exceeding 150. This suggests that using a larger window with a moderately small step allows the model to capture sufficient temporal patterns while still providing a diverse training set through overlapping windows. In contrast, using a step size equal to or greater than the window reduces data utilization, possibly leading to underfitting or insufficient learning.

2. (Approximately 200 words.)

(i) (15 points) Include 'Volume' as an additional input feature in your model. Discuss the impact of incorporating 'Volume' on the model's performance.

(ii) (15 points) Explore and report on the best combination of input features that yields the best MSE. Briefly describe the reasons of your attempts and analyze the final, optimal input combination.

(i) Model Setting: hidden_dim=500, num_layers=1, lr=0.001, epochs=100, unnormalized training data, features=(Open, High, Low, Close, Volume)

Window Size	Step Size	Test MSE (Raw Input)
10	5	1105.13
10	10	1378.38
5	10	1365.30

To evaluate the effect of incorporating trading volume, I compared the model's performance using the same configuration (window size, step) with and without the 'Volume' feature. The three MSE results all suggest that adding volume may have introduced noise or irrelevant variance that distracted the model from learning useful price patterns. This suggests that volume may not provide direct predictive value for forecasting next-day high prices under the current model architecture and preprocessing pipeline. Therefore, simply adding more features does not guarantee better performance; feature relevance and data scale consistency are crucial in time-series modeling.

(ii) Model Setting: hidden_dim=500, num_layers=1, lr=0.001, epochs=100, normalized training data, multiple feature combinations

Feature Set	Test MSE (Normalized Input)
Close	0.997
Open, Close	**0.776**
Close, Volume	0.840
High, Low, Close	1.385
Open, High, Low, Close	1.125
Open, High, Low, Close, Volume	0.805

I compared six different input feature sets using (window size, step)=(10, 5). The best performance (MSE = 0.776) came from using just 'Open' and 'Close' prices, suggesting that these two features contain the most predictive power for the next day’s high. Interestingly, adding 'Volume' or using all five features did not lead to improvement, and some combinations even degraded performance. This highlights the importance of careful feature selection in time-series forecasting tasks.

3. (15 points) Analyze the performance of the model with and without normalized inputs in Lab 4. You can use experimental results or external references (which must be cited) to support your conclusions on whether normalization improves the model's performance. (Approximately 100 words.)

Model Setting: hidden_dim=500, num_layers=1, lr=0.001, epochs=100, **normalized training data**, features=(Open, High, Low, Close)

Window Size	Step Size	Test MSE (Normalized Input)
10	5	0.76
10	10	1.42
5	10	2.60

Compared to the results in Q1 using raw input, normalization significantly improved model performance across all parameter combinations. For the (10, 5) setting, the MSE dropped from 17.67 (raw) to 0.76 (normalized). This demonstrates that scaling input features helps the LSTM model converge more effectively by stabilizing gradients and improving numerical consistency. Moreover, the performance gap widens as step size increases, indicating that normalized input is especially helpful when training samples are fewer or more spaced.

4. (10 points) Why should the window size be less than the step size in Lab 4? Do you think this is correct? If you use external sources, please include references to support your response. (Approximately 50 words.)

According to results in Q1 and Q3, it shows that window < step leads to much worse MSE in both raw and normalized settings. Overlapping windows increase training samples and allow the model to better capture local temporal patterns. Without sufficient overlap, the model may underfit due to reduced data utilization and contextual learning.

5. (15 points) Describe one method for data augmentation specifically applicable to time-series data. Cite references to support your findings. (Approximately 100 words.)

One effective method for time-series data augmentation is window slicing, where overlapping subsequences are extracted from the original time series to create more training samples. This increases data diversity, especially when labeled data is limited, and improves model generalization. It is particularly useful in scenarios like financial forecasting and sensor analysis. By adjusting the window and step size, we can control the granularity and overlap of training instances. This technique has been widely adopted in the literature *(Um et al., 2017; Fawaz et al., 2019)* and is simple yet powerful for enhancing LSTM and CNN performance on time-series tasks.

6. Discuss how to handle window size during inference in different model architectures (approximately 150 words):
   (i) (5 points) Convolution-based models
   (ii) (5 points) Recurrent-based models
   (iii) (5 points) Transformer-based models

(i) Convolution-based models:

Convolutional models process fixed-size windows using sliding kernels. During inference, the input is typically segmented into windows of the same size used in training to maintain feature consistency. Larger inputs can be processed using overlapping or strided windows to enable dense prediction over time.

(ii) Recurrent-based models:

Recurrent models like LSTMs require sequential input and inherently preserve temporal order. At inference, they consume input sequences of the same window size used during training. Alternatively, they can process streaming data step by step in an online fashion, maintaining hidden states across time.

(iii) Transformer-based models:

Transformers can handle variable-length sequences but rely on position encoding to retain temporal structure. For consistency, inference should use the same window length as training. In longer sequences, a sliding window or causal masking approach can be applied to ensure efficient attention and prevent data leakage.