National Tsing Hua University

11320IEEM 513600

Deep Learning and Industrial Applications

Homework 4

Name: 黃皓澤 Student ID: 113034519

1.

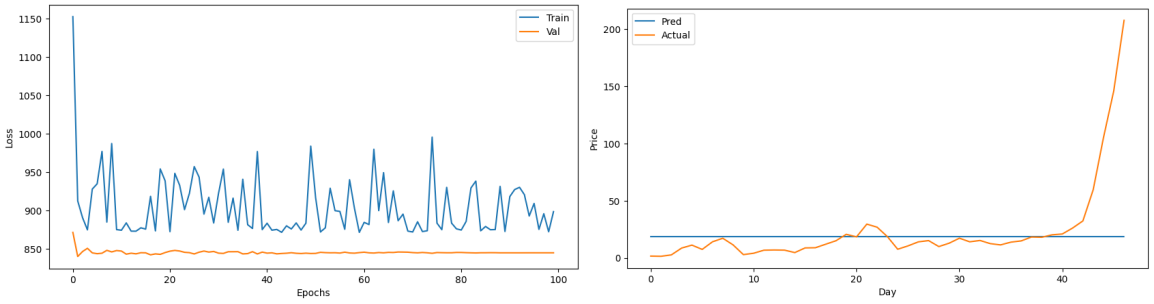
window sizes	steps	MSE
10	15	350.0246095846937
5	10	136.40854742106686
5	5	15.086855883519174

The table shows that as both window_size and step decrease, the model's performance improves significantly, as indicated by the declining MSE. A smaller window_size (5) likely captures more relevant short-term patterns, while a smaller step (5) increases the number of training samples, leading to better learning. The best performance is achieved with window_size=5 and step=5 (MSE ≈ 15.09), suggesting that the model benefits from more granular and overlapping sequences. In contrast, a larger window or step may result in fewer or less focused training samples, causing the model to generalize poorly and produce higher prediction errors.

2.

(i)

Adding 'Volume' as a feature caused the MSE to surge to 376.714, indicating worse performance. 'Volume' may introduce noise because its patterns differ from price movements and are harder for the model to interpret. Instead of helping, it likely confused the model, making learning less effective. This shows that adding irrelevant or unstable features can degrade prediction accuracy rather than improving it.



window_size=5, step=10

(ii)

Combination	MAE
'Open', 'High', 'Low', 'Close'	15.086855883519174
'Open', 'Close', 'Volume'	1095.0659712097886
'Open', 'Close'	20.414907266384798

window sizes: 5, steps: 5

The best feature combination is 'Open', 'High', 'Low', 'Close', yielding the lowest MAE of 15.09. This set captures comprehensive daily price information, giving the model a richer understanding of market dynamics. Attempts using 'Volume' caused performance to drop significantly (MAE ≈ 1095.07), likely due to its weak correlation with next-day prices. Simpler inputs like 'Open', 'Close' performed better than noisy features but worse than full data. Thus, complete price data is optimal.

3.

without normalized	15.086855883519174
MinMaxScaler normalized	0.00013238007726618844

window sizes: 5, steps: 5

Experimental results demonstrate that applying MinMaxScaler normalization dramatically improves model performance, reducing the MSE from 15.09 to 0.00013. This significant enhancement is attributed to normalization aligning feature scales, which facilitates more efficient learning and faster convergence in gradient-based models like LSTM. Normalization ensures that each feature contributes proportionally, preventing features with larger scales from dominating the learning process. This aligns with established practices in machine learning, where normalization is crucial for models sensitive to feature scales.

4.

Using a window size smaller than or equal to the step size reduces overlap between sequences, minimizing redundancy and overfitting. It also improves training efficiency. Excessive overlap can lead to data leakage and poor generalization. Thus, this practice is often correct and effective.

5.

One effective method for time-series data augmentation is time warping, which stretches or compresses segments of the signal to create variations while preserving trends. This technique helps models generalize better by simulating natural speed fluctuations in real-world data (Um et al., 2017). Another approach is jittering, where small Gaussian noise is added to the original signal, enhancing robustness to measurement errors (Wen et al., 2021). Both methods are simple yet improve model performance, particularly in applications like healthcare monitoring and industrial sensor analysis. Time warping is especially useful for preserving long-term dependencies, while jittering helps with noise resilience.

References:

Um et al. (2017). Data augmentation of wearable sensor data for Parkinson's disease monitoring.

Wen et al. (2021). Time-series data augmentation for deep learning.

6.

(i) Convolution-based models

They use a fixed receptive field via kernel size. At inference, keep the last window_size steps in a rolling buffer, apply causal convolutions, and slide forward so outputs depend only on past inputs.

(ii) Recurrent-based models

Use RNNs, LSTMs or GRUs which carry a hidden state. For inference, either warm up the state with the last window_size samples or cache the hidden state and feed only the new sample each step. Caching avoids reprocessing the full window, improving efficiency.

(iii) Transformer-based models

Standard Transformers attend over a fixed context. At inference, maintain a sliding context buffer of size window_size and cache past key/value pairs so each new token attends only to recent positions. Variants like Longformer or Transformer-XL extend context with sliding-window or recurrence mechanisms without quadratic cost.