Deep Learning for Industrial Applications

113034504 何懿城

1. 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.

Window, Step	Dataset Size	Train MSE	Val MSE	Test MSE
10, 15	227, 57, 31	244.3047	329.7342	588.6926
8, 10	340, 86, 47	97.0410	212.4413	303.7918
5, 5	681, 171, 94	5.7430	7.9991	16.1738

- Hyperparameter (All experiment adopt this setting)
  - Learning rate: 10^{-3}
  - Epochs: 100
  - Hidden Layers: 2
  - Hidden Units: 500
- From above table, the experiment with window 5 and step 5 has the best performance among three attempts, I think the main reason is this attempt with largest data size to train model.
- And for the other 2 experiments, the train, valid mean square errors are much larger than previous experiments which shows that the 2 experiments might be underfitting. The reason of worse performance may be the data size is not enough for model learning longer time relationship.

2. Approximately 200 words
   - Include 'Volume' as an additional input feature in your model. Discuss the impact of incorporating 'Volume' on the model's performance.
     - After including ‘Volume’ as an additional input feature, the performance of model becomes much worse than the model with only 4 features (Open, High, Low, Close)
     - The reason might be the scale of Volume is much larger than and different from the other features, and this is difficult for LSTM to fit the data.
   - 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.

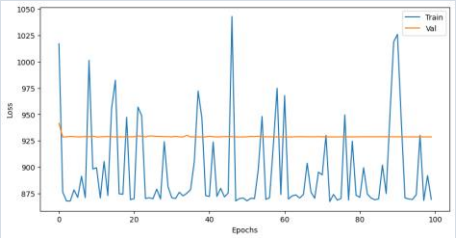
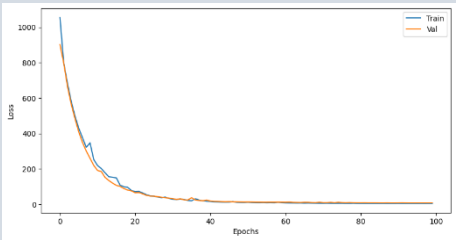
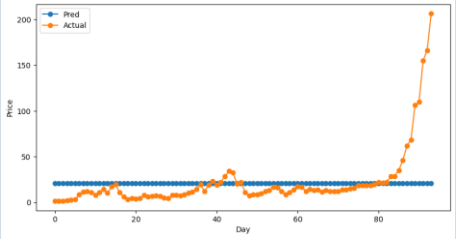
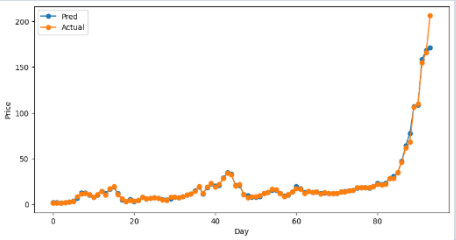
Feature	Train MSE	Val MSE	Test MSE
Open, High, Low, Close, Volume	873.0899	928.6604	1093.9988
Open, High, Low, Close	5.7430	7.9991	16.1738
Open, High, Low	5.3752	6.9712	15.1605
High, Low, Close	5.1322	6.7612	14.7531
Open, High, Close	5.1832	7.3086	15.2431
Open, High	4.6594	6.0660	14.9059
High, Low	4.6146	6.3685	13.8915
High, Close	4.8124	5.9478	14.6322
High	5.3180	6.5583	14.9270

- ➢ My experiment strategy was to keep 'High' as a mandatory feature and eliminate one feature at a time based on validation MSE performance
- ➢ From the results, the model using only High and Low achieved the best overall performance, while the model with two features (High, Close) ranked second
- ➢ A noticeable trend was that validation MSE generally decreased as the number of features reduced, suggesting that removing redundant inputs helped the model generalize better. However, when using only High alone, the performance worsened, indicating that High by itself lacks enough information for accurate prediction. Therefore, although simplifying the feature set improves generalization, it is important to retain complementary features like Close to maintain sufficient predictive power

3. 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.
   - — I choose the model with 5 (High, Low, Open, Close, Volume) features to implement the with and without normalized inputs experiments.

Model	Normalized Inputs	Train MSE	Val MSE	Test MSE
LSTM (5 features)	N	873.0899	928.6604	1093.9988
LSTM (5 features)	Y	5.4986	8.9463	14.5987

— Training and Validation Fitting Plot and Testing True / Pred value Plot

Plot Category	Without Normalization	With Normalization
Training / Val **Performance**		
Testing True / Pred		

— From the result table, we can see that model with normalization has better training, validation and testing MSE.

— From the fitting plot, we can see that model with normalization converges faster and more stable compared to the model without normalization.

— The above suggests that normalization scaling features to similar range helps the model learn more effectively and improves the model performance and training efficiency.

4. 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.
    - I think window size should be greater than or equal to step size for LSTM, otherwise, we will skip data and lose temporal information. Hence, I think that having window size smaller than step size is inappropriate.
5. Describe one method for data augmentation specifically applicable to time-series data. Cite references to support your findings. (Approximately 100 words.)
    - Time Warping
        - Randomly stretching or compressing segments of the time series to create new variations while preserving the overall shape and patterns. This technique helps models generalize better by simulating realistic temporal distortions that might occur in real-world data.
        - Time warping has been successfully applied in various fields like sensor data analysis and healthcare monitoring. Um et al. (2017) demonstrated its effectiveness for enhancing deep learning models on physiological time-series data.
        - Reference: https://arxiv.org/abs/1706.00527
6. Discuss how to handle window size during inference in different model architectures (approximately 150 words):
    - Convolution-based models
        - convolutional models typically use a fixed window size based on the receptive field defined during training. The input time-series is segmented into overlapping or non-overlapping windows matching this size. If the sequence is longer, a sliding window approach can be used to process the entire input; if shorter, padding is applied.
    - Recurrent-based models
        - Recurrent models like LSTMs or GRUs can handle variable-length sequences during inference, but the window size must be compatible with what the model saw during training for stable performance. Long sequences can be processed step-by-step or truncated, while short ones might require padding.
    - Transformer-based models
        - Transformer models usually expect fixed-length inputs due to their positional encoding. During inference, longer sequences are either split into fixed-size windows or use memory-efficient variants (like Longformer) to handle full sequences. Short sequences are padded, ensuring consistency with training conditions.