AI6103 Deep Learning and Applications
Final Project Report

Informer and Applications

SD6101 Project Assignment Submission: Informer Beyond Efficient Transformer (github.com)

Zhao Yuhan	Wang Qiyue	Wang Yutian	Eugene Ho Hong Zhuang	Koo Chia Wei
G2204624C	G2204608F	G2204884F	G2204889B	G2202810D
zhao0431@e.ntu.edu.sg	wang1901@e.ntu.edu.sg	wang1910@e.ntu.edu.sg	eho010@e.ntu.edu.sg	ckoo004@e.ntu.edu.sg

Abstract

In this project, we choose to reproduce the 2021 AAAI best paper, Informer as the research object. Informer is a prediction model based on the Transformer structure for long-term sequences. Informer uses methods such as multi-head self-attention mechanism, depth wise separable convolution, periodic attention mechanism, and stacked structure, which can enhance the robustness of the model and integrate Global and key local information, and consider different time scales, to achieve effective modeling and forecasting of long sequences.

After carefully studying the paper and related work, we reproduced the entire work based on our own understanding, dis-cussed the impact of different optimizers and loss functions on the model on the original data set, and tested one brand new dataset Singapore Exchange Limited (S68.SI) to explore In-former Generalization ability and existing problems.

1. Introduction

The development of time series prediction methods has a long history. In prediction problems, there are several important issues that need to be addressed, including prediction accuracy, input sequence length, time complexity, space complexity, predictability length, interpretability, and so on. From the ARIMA models to deep learning models such as RNN and LSTM, solutions have been proposed for time series prediction problems. The application of transformer models, which were originally designed for natural language processing, in time series prediction has led to new and updated solutions to address several prediction issues. With the development of technology and changes in data, longer input and output sequence lengths have become increasingly necessary. As a key solution to this issue, In-former has attracted our attention.

2. Background and Related Work

2.1. Development of Transformer

The Transformer model, originated from Google Brain team's "Attention is All You Need" in 2017, utilized self-attention mechanisms to model sequences and achieved remarkable performance in machine translation tasks. Its development was built upon the neural machine translation model based on attention mechanisms proposed by Bahdanau et al. in 2014 and the CNN-based Seq2Seq model proposed by Kalchbrenner and Blunsom in 2015. In 2018, Facebook AI research team introduced the Bert model, which learned rich language representations through pre-training language models and achieved top performance in various downstream tasks. Similarly, in 2019, Google introduced the T5 model, which is a universal text generation model applicable for various NLP tasks. The Transformer model's progress in natural language processing provided novel ideas and methods for the application of deep learning in other fields. This paper applied the Transformer model in long-term sequence prediction in 2021 and modified the attention mechanism to overcome Transformer's shortcomings in temporal data.

2.2. Development of Informer

Based on informer, there are several types of X-former models come up like Seformer(2022), which is based on bi-nary position encoding, has optimized the loss of information during the transmission process. And then, a multi-granularity attention head mechanism named as muformer((2022)) to make each attention head focus on specific information and improve the efficiency. And also, a triangular, variable-specific attention named triformer(2022) to improve accuracy and efficiency without increasing the sources used.

3. Informer

Informer is a model promoted in 2021 to improve the shortcomings of traditional Transformer model in view of the problem of long sequence time-series forecasting. Over-all, the Informer model still preserves the Encoder-Decoder architecture as shown in Figure 1.

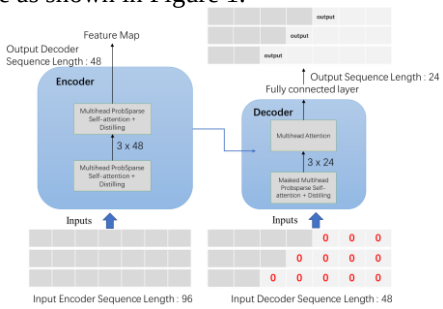

Figure 1: Architecture of Informer.

The encoder is shown on the left in Figure 1.The process is that the encoder receives long sequence input, and the feature map is obtained by the Prob-Sparse self-attention module as well as the self-attention distilling module. And the decoding process is shown on the right: the decoder receives the long sequence input, pads the target elements into zero, interacts with the encoding features through multi-head attention, and finally predicts the whole output target part directly.

3.1. Prob-Sparse Self-attention

First, review the classic self-attention mechanism: Calculate the input's attention matrix using the scaled dot product from three inputs (query, key, value).

$$A(Q,K,V) = Softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$$

And the probability form of the attention coefficient of the Query i is:

$$A(q_i,K,V) = \sum_j \frac{k(q_i,K_j)}{\sum_l k(q_i,K_l)} V_j = E_p(k_j|q_i)[V_j]$$

In this paper, the sparsity of the classical self-attention mechanism is verified by experiments, namely the long-tail distribution phenomenon of self-attention feature map. As shown in Figure 2, very few dot products contribute most of the attention score, which means that the rest can be ignored.

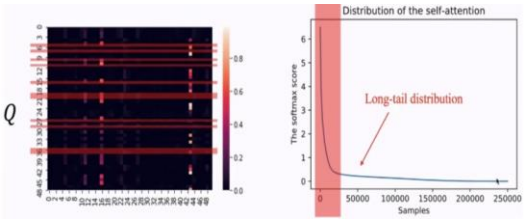

Figure 2: Self-Attention Distribution.

In other words, a large part of 'Q' is not Active Query which is strongly correlated with 'K', but Lazy Query which is close to uniform distribution as shown in Figure 3.

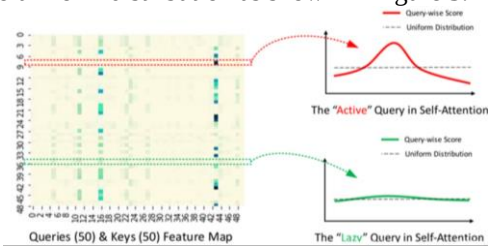

Figure 3: Active and Lazy Query.

In order to measure the sparsity of query, KL divergence is used to calculate the relative entropy of the probability distribution of attention of query and the probability distribution of uniform distribution. The sparsity evaluation formula of the i[th] query is as follows:

$$\mathrm{M}(q_i,K) = ln\sum_{j=1}^{L_k} e^{\frac{q_i k_j^T}{\sqrt{d}}} - \frac{1}{L_k}\sum_{j=1}^{L_k} e^{\frac{q_i k_j^T}{\sqrt{d}}}$$

The first term is Log-Sum-Exp (LSE) for all the keys, and the second term is their arithmetic average. But there are two problems with this formula:

- The computational complexity of dot product pairs is $O(L_Q L_k)$
- The LSE calculation may be unstable, which may result in data overflow and error reporting.

In order to solve these two problems, the following measures are adopted:

1. Random sample dot product pairs to compute $\mathrm{M}(q_i,K)$
2. Replace $ln\sum_{j=1}^{L_k} e^{\frac{q_i k_j^T}{\sqrt{d}}}$ with $\max\left(\frac{q_i k_j^T}{\sqrt{d}}\right)$

Finally, according to the above evaluation method, the formula of Prob-Sparse self-attention can be obtained:

$$A(Q,K,V) = Softmax(\frac{\bar{Q}K^T}{\sqrt{d}})V$$

The Prob-Sparse self-attention allows each key to only attend to the u dominant queries, that is the selected. This approach solves the first problem in Transformer, namely, the quadratic computational complexity of the self-attention mechanism.

3.2. Handling of Lazy queries

In the above process, only the vectors of the points corresponding to active queries are obtained after attention coding. And it is clear that the probability distribution of Lazy queries is close to uniform distribution, so their vectors calculated by self-attention should also be close to the average of all V vectors. Therefore, for the vectors corresponding to

the remaining lazy queries, we replace them with the average of the V vectors. In this way, the vectors for all the time points are acquired.

3.3. Encoder Distilling

The goal of the encoder is to halve the time dimension of a single level feature by means of an attentional distilling mechanism, allowing the encoder to process a longer sequence of inputs in the case of limited memory. As the result of the ProbSparse self-attention mechanism, the encoder's feature map has a redundant combination of values V. Here the distilling operation is used to give higher weight to the dominant features with dominant attention and generate the focus self-attention feature map at the next layer. The Encoder Stack proposed in this paper is actually a combination of multiple encoders and distilling layers.

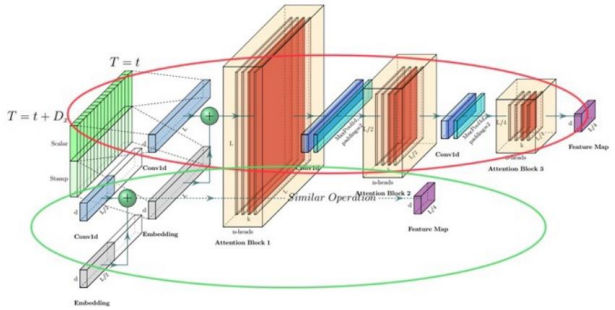

Figure 4: Encoder Stack.

Figure 4 illustrates the basic structure of the encoder stack. The red circle represents the first and primary stack, which receives the entire input sequence; The green circle below it is the second stack. The input sequence for the main stack is halved and then used as input for the second stack. The process from j to j+1 level distilling operation formula is as follows:

$$X_{j+1}^{t} = MaxPool(ELU(CONV1d((X_j^t)_{AB})))$$

Conv1d represents a one-dimensional convolution operation on the time series. ELU is an activation function, with a max-pooling layer (stride=2), and half of the samples will be down-sampled every time. This saves on memory usage. In order to enhance the robustness of the attention distilling mechanism, multiple half-length copies of the main sequence are also constructed, each of which is half the length of the previous copy. These copy sequences and the main sequence are subjected to the same attention distilling mechanism to construct multiple L/4 length feature maps. Finally, these feature maps are combined to form the final feature map of the input encoder.

Through the above methods, the size of the feature map can be reduced step by step without consuming too much memory in computing space, which solves the second problem of Transformer: high memory usage.

3.4. Generative Style Decoder

The decoder is designed to predict all outputs of a long sequence through a single forward calculation. Basically, Decoder is composed of two decoder layers, and the interior of each Decoder layer includes: a mask self-attention; A multi cross-attention that is responsible for the interaction of target sequence and source sequence.

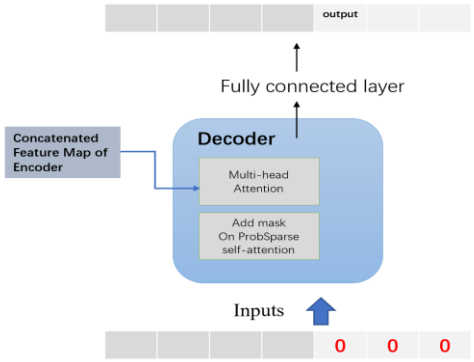

Figure 5: Decoder Structure.

As shown in Figure 5 and Figure 6, the decoder input is composed of two parts: the first is the output of the encoder and the second is the input of the decoder after the embedding(masked by 0).

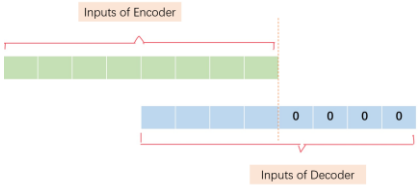

Figure 6: Inputs of Encoder and Decoder.

The purpose of the entire model is to predict the exact value of the masked part. The start token of the decoder in the informer model samples a short sequence (the previous fragment of the sequence to be predicted) from the input. The input to the decoder is:

$$X_{feed\_de}^{t} = Concat(x_{token}^{t}, x_0^t) \in R^{L_{token} \times d_{model}}$$

Where x_0^t represents a placeholder (predicted value); x_{token}^t for start character. Part of the sequence is dynamically sampled from the input sequence close to the predicted target as a "starting Token".

The Decoder in Informer is different from the traditional Decoder mainly in output style. As Figure 7 shows, the traditional Transformer follows a step by step style, putting the output of the previous step into the decoder to get the output of the next step. Therefore, only one time step can be output for once.

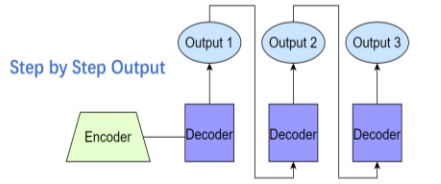

Figure 7: Traditional Decoder Output Style.

Informer proposes a one-step prediction process as illustrated in Figure 8.

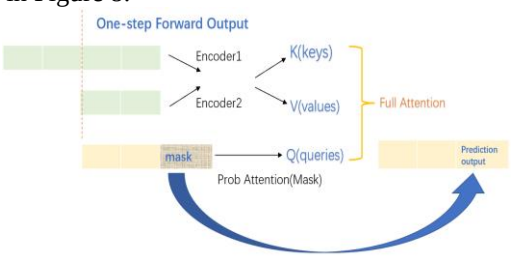

Figure 8: Long Sequential Outputs Through One Forward Procedure.

4. Experiments and Results

4.1. WTH Experiment

WTH: The WTH data set is a meteorological data set. The data set consists of 12 items of weather data such as temperature, humidity, and wind speed in 1,600 locations in the United States during the four years from 2010 to 2013. It collected data every hour, with a total of 35,064 pieces of data. Therefore, the dimension of the data set is [35064, 12]. In the experiment, we use "Wet Bulb Celsius" as the target value, and the remaining eleven indicators as the fea-tures.
Data and characteristics: (1) The data are all floating-point types, and most of the data are normally distributed, but there are some outliers and deviations. (2) Some indicators, such as temperature and dew point temperature, wind speed, and wind direction, have strong correlations, which may be used as a reference for my surname training and feature extraction. (3) Some indicators, such as air pressure and sea level air pressure, have certain missing values, which need to be processed.

Training Details

Evaluation criteria: For each prediction result, we use two evaluation indicators, MSE and MAE.
Platform: All models are trained/tested on a single Nvidia V100 32GB GPU.
Initial setting: Divide all data into a training set, verification set, and test set according to the ratio of 7:1:2, batch size 32, using 8-head attention mechanism, Gelu. We set the initial learning rate to 0.0001, after each epoch, the learning rate is halved. A total of 6 epochs are trained and iterated twice. In this experiment, we stacked three layers of encoders in the Informerstack model. Additional details about the Informer models are given in Table 1.

Table 1: The Informer model components details .

Encoder:		
Inputs	Conv 1d: 1×3	Embedding $d = 512$
Prob-attention Self-attention	Muti-head Prob-attention $h = 16, d = 32$	
	Normalized Layer, Dropout $p = 0.1$	
	FFN $d = 2048$, GELU-active	
	Normalized Layer, Dropout $p = 0.1$	
Distilling	Conv 1d: 1×3 , ELU-active	
	Max-pooling stride=2	
Decoder:		
Inputs	Conv 1d: 1×3	Embedding $d = 512$
Prob-Mask attention	Add Mask on attention block	
Self-attention	Muti-head Prob-attention $h = 8, d = 64$	
	Normalized Layer, Dropout $p = 0.1$	
	FFN $d = 2048$, GELU-active	
	Normalized Layer, Dropout $p = 0.1$	
Final:		
Output	FCN $d = d_{out}$	

Note: We conducted 5 experiments, each of which randomly selected the training set and validation set, and kept 10% of the validation set, and then averaged the results of these 5 experiments to obtain more reliable evaluation results.

Results and Analysis

Table 2 summarizes the test results of testing the Informer and Informerstack models on the WTH dataset and applying different optimizers and loss functions.
In the original paper, the author chose MSE, that is, L2 loss as the loss function, and used the Adam optimizer. The purpose of this experiment is to test whether the loss function and the optimizer can achieve better performance on this data set. The loss function experiment group chooses MAE (L1 loss) and HuberL1 loss (Huber, 1992), and the optimizer chooses Adammax (Luo, et al., 2019).
From the results, we can see that the test result with L1 loss function has a larger MSE but a smaller MAE. This is because, as shown in the data analysis, there are certain samples in the data that are strongly correlated with each other, as well as samples that are similar but not quite equal. In this case, using the L1 loss function may cause the error term of some samples to be ignored, resulting in a larger value of MSE. However, MAE pays more attention to the absolute value of the sampling error, so it is small.

Table 2: Test results of WTH dataset on different models, loss functions and optimizers.

Models	Informer										Informerstack					
Optimizers	Adam						Adammax				Adam		Adammax			
Loss Function	MSE		L1 loss		HuberL1		MSE		HuberL1		MSE		MSE		HuberL1	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
WTH	0.311	0.366	0.326	0.344	**0.320**	**0.356**	0.305	0.360	**0.303**	**0.346**	0.319	0.363	0.326	0.373	0.317	0.356

The MSE of the HuberL1 loss function is the smallest, and although the MAE is slightly larger than that of the L1 loss function, it is still smaller than the original setting. This is because there are still some outliers in the sample. At this time, the L2 loss function will perform better, and Huber-Loss combines the advantages of L1 and L2. When the error of the loss function is small, it uses MSE as the loss function. Currently, it is sensitive to outliers, but when the error is large, it uses MAE as a loss function, which is less sensitive to outliers currently. In summary, we chose the HuberL1 loss function for the next experiment.

In the test of the optimizer Adammax, regardless of using the MSE (L2) loss function or the Huberl1 loss function, the resulting MSE and MAE indicators are lower than those of the Adam optimizer, indicating that the effect is better.

Finally, we tested Informerstack and found that on this data set, the performance of Informerstack and Informer was not much different. After choosing the better loss function HuberL1 and the optimizer Adammax, the performance of Informerstack was worse, as follows for Two reasons: (1) The data set is relatively simple (2) The long-term dependencies in the data set are weak. Informerstack adds additional network layers and parameters, and has a stronger ability to capture long-term dependencies, so it is simple in processing and long-term When relying on relatively weak datasets, Informerstack will overfit, leading to poor training results.

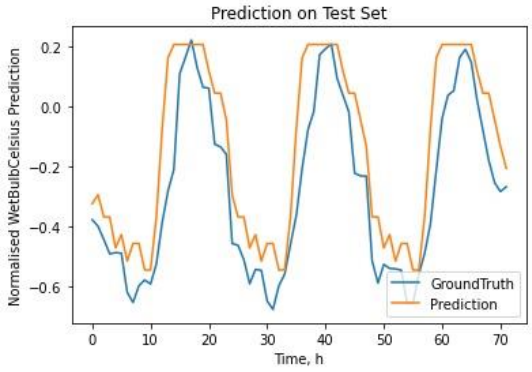

Figure 9: Forecasting on Test Set using Adammax optimizer and HuberL1 loss function.

4.2. Global Land and Ocean Temperature Forecasting

Global Climate Change Data: The Global Climate Change Data is a time series dataset that contains historical temperature data for the Earth, starting from 1850 for maximum and minimum land temperatures, and global ocean and land temperatures, and 1750 for average land temperature. The dataset is based on raw data from Berkeley Earth, provides high-resolution land and ocean time series data and gridded temperature data. It includes variables such as LandAverageTemperature, LandMaxTemperature, LandMinTemperature, LandAndOceanAverageTemperature, with their uncertainties. Only data from 1850 onwards is used due to missing data in the earlier years. The data can be used for various purposes such as analyzing global temperature trends and investigating climate change effects.

Training Details

Here, we utilized the informer to forecast the average global land and ocean temperature using monthly data from 1850 to 2015, after removing all empty entries. This dataset was acquired from data.world and generated by Berkeley Earth, and includes multiple variables for each recorded month, such as LandAverageTemperature, LandAverageTemperatureUncertainty, LandMaxTemperature, LandMaxTem-peratureUncertainty, LandMinTemperature, LandMin-TemperatureUncertainty, LandAndOceanAverageTemper-ature, and LandAndOceanAverageTemperatureUncertain-ty. All uncertainties were assessed using a 95% confidence interval for each respective variable and were relabeled as GT.csv. The dataset was subsequently divided into a 0.6:0.2:0.2 ratio for the training, validation, and testing sets, respectively.

Results and Analysis

A series of experiments were conducted to predict global land and ocean temperature trends for various months using different hyperparameters and settings as outlined in Table 3. The findings revealed that lowering the learning rate decay significantly for longer epochs resulted in an increase in validation loss, as shown in Figure 10, which indicated an occurrence of overfitting. However, despite having lower test errors for mean squared error (MSE) and mean absolute error (MAE) at the end of Experiment 2's training than Ex-

periment 1's training, Experiment 2's training achieved substantially lower validation loss in the early epochs (~7), suggesting benefits to reducing the learning rate decay.
Moreover, the reduction of sequence, label, and prediction length resulted in significant reductions in both training and validation losses, which were also reflected in the test errors for MSE and MAE, where shorter sequences led to better predictions. A similar pattern was observed in the training and validation loss trends.
Furthermore, experimenting with early stoppage revealed that training with very few epochs in Experiment 6 produced test errors for MSE and MAE comparable to those obtained after training for 100 epochs. Throughout all experiments, it was observed that the training and validation loss decreased significantly during the early epochs (<15) followed by a plateau or an increase in validation loss. These observations suggest that there may not be significant improvements in performance by increasing the number of training epochs, and the returns on performance diminish as the number of epochs increase.

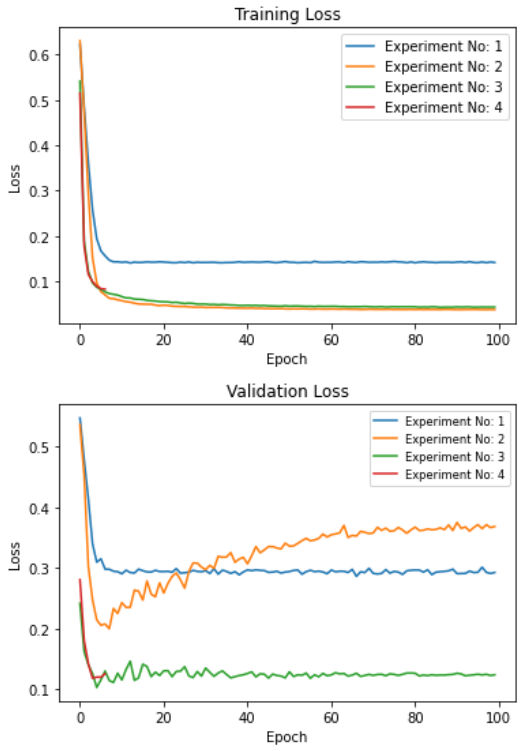

Figure 10: Train and Validation Loss on Global Land and Ocean Temperature.

By adopting the parameters from Experiment 5, the Informer was able to forecast the trend of the Global Land and Ocean Average Temperature relatively well as shown in Figure 11.

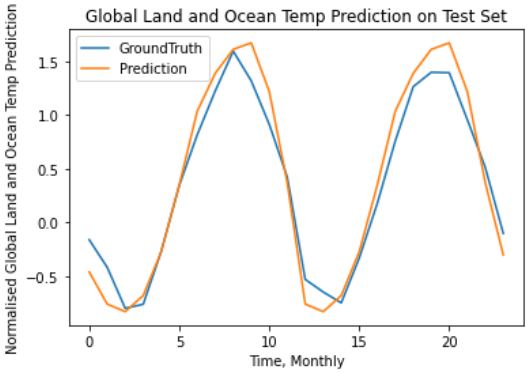

Figure 11: Forecasting on Test Set using Experiment 4's parameters and Settings.

Table 3: Test Results of Global Land and Ocean Average Temperature with different Hyperparameters and Settings.

Experiment No.	Epoch	Learning Rate Decay per Epoch	Sequence, Label, Prediction Length	Test Error, MSE	Test Error, MAE
1	100	0.5 (50%)	[96,48,24]	0.401	0.493
2	100	0.05 (5%)	[96,48,24]	0.302	0.391
3	100	0.05 (5%)	[48,24,12]	0.141	0.288
4	10*	0.05 (5%)	[48,24,12]	0.165	0.313

** Early Stoppage*

4.3. Stock Forecasting on S68.SI

S68.SI Stock Data: The dataset contains historical data of the S68.SI index, which represents the stock for Singapore Exchange Limited traded on the Singapore Stock Exchange. The data includes Open, High, Low, and Close prices from January 1, 2011, to December 31, 2017, in Singapore dollars. The data can be used to analyze the historical performance of the stock and to identify trends and patterns for making informed investment decisions.

Training Details

Limited, represented by S68.SI, a stock exchange, using data from November 23, 2000, to end of 2017. To account for unforeseen circumstances, such as a financial crisis, we removed certain data points from the dataset since there were no variables that indicated the occurrence of a crisis. The available variables recorded per day are limited to the highest transaction price (High), lowest transaction price (Low), open market price (Open), close market price (Close), and transaction volume (Volume), and was relabeled as S68_4.csv.
We excluded the adjusted close market price (Adj Close) as it is highly correlated with the close market price, making it redundant. Additionally, the transaction volume did not provide information on selling and buying transactions, hence it was also removed from our analysis. We aim to predict

trading swing characteristics in the market by holding positions for one day to a couple of weeks, which focuses on capturing short-term price fluctuations based on technical analysis, rather than considering long-term factors such as company growth, products, or other distinct financial indicators.

Results and Analysis

Our previous experiment showed that increasing the number of training epochs for the informer model did not result in significant performance improvements. Therefore, we attempted to predict market swings by modifying sequence lengths, label lengths, and prediction lengths with an early stoppage mechanism. While we achieved relatively low test errors on metrics such as MSE and MAE by shortening these parameters, Table 4 revealed that the predicted trend was inaccurate. The model was unable to predict the market trend accurately based solely on the Open, Close, High, and Low prices of the market, highlighting the need to consider additional financial indicators for accurate market trend predictions. To validate our findings, we compared the performance of models trained for 100 epochs in Experiment 4 and found that the trend of the prediction was similarly inaccurate, as shown in Figure 12. Thus, we conclude that the informer model can only forecast market swings to a certain extent based on these limited financial indicators.

Table 4: Informers on Stock Price Experiments and Hyperparameters Tunning.

Experiment No.	Epoch	Learning Rate Decay per Epoch	Sequence, Label, Prediction Length	Test Error, MSE	Test Error, MAE
1	5*	0.05 (5%)	[96,48,24]	0.168	0.339
2	10*	0.05 (5%)	[48,24,12]	0.0934	0.237
3	13*	0.05 (5%)	[24,12,6]	0.0379	0.152
4	**100**	**0.05 (5 %)**	**[24,12,6]**	**0.0371**	**0.151**

** Early Stoppage, Patience = 3*

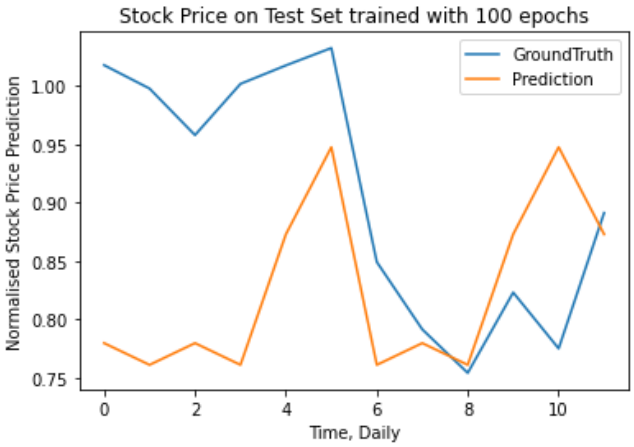

Figure 12: Stock Price on Test Set Trained with 100 epochs.

5. Discussion

5.1. Outliers

The quality of data can have a significant impact on the performance of the informer model. Inaccurate, incomplete, or biased data can result in poor predictions and unreliable models. The data used in predicting market trends may not capture all relevant market factors, such as the impact of financial crises, which can lead to inaccurate predictions. During periods of financial crises, markets can exhibit unusual behavior, characterized by increased volatility, sharp declines in asset prices, and heightened uncertainty, which can be difficult to predict using traditional financial indicators.

In general, outliers can also significantly affect the performance of the informer model. Outliers are data points that deviate significantly from the rest of the data, and they can distort the model's predictions by skewing the statistical distribution. Outliers can occur due to errors in data collection or processing, natural variations in the data, or extreme events. If not identified and properly handled, outliers can result in models that overfit or underfit the data, leading to poor predictions and unreliable results.

To mitigate the impact of outliers and improve model performance, it is important to carefully pre-process the data by identifying and removing outliers or imputing missing values. Additionally, incorporating domain-specific knowledge and additional relevant data sources can help to improve the quality of the input data and increase the accuracy of the model's predictions.

6. Conclusion

Based on the findings of the experiments, it can be concluded that the informer model can be used to predict the trend of the Global Land and Ocean Average Temperature relatively well, by reducing the sequence, label, and prediction length, and by decreasing the learning rate decay to a certain extent. However, increasing the number of training epochs does not necessarily result in significant performance improvements, and overfitting can occur if the learning rate decay is set too low.

On the other hand, the model's ability to predict market swings is limited when based solely on the Open, Close, High, and Low prices of the market, as shown by the inaccurate trend predictions in both short and long training epochs. This highlights the need to consider additional financial indicators to accurately predict market trends.

Overall, the In-former model can be a useful tool for time series prediction, but careful consideration of hyperparameters and additional data sources is necessary to ensure accurate results.

Based on the experiments conducted with the Informer model, it was found that the performance of the model was affected by various hyperparameters, including the number of training epochs, learning rate, and sequence length.

The experiments revealed that increasing the number of training epochs did not always result in significant performance improvements. Instead, the returns on performance diminished as the number of epochs increased, and the model tended to plateau or experience an increase in validation loss after the early epochs. Therefore, it may not always be necessary to train the model for a large number of epochs, and early stoppage mechanisms can be used to achieve similar or even better results.

Lowering the learning rate decay significantly for longer epochs was found to result in overfitting, as reflected in an increase in validation loss. Therefore, it is important to find an appropriate learning rate decay to avoid overfitting and achieve better generalization performance.

The reduction of sequence, label, and prediction length was found to lead to significant reductions in both training and validation losses, resulting in better predictions. Shorter sequences were found to lead to better predictions, indicating that the model benefits from being fed smaller chunks of data.

In conclusion, the performance of the Informer model can be significantly influenced by various hyperparameters, including the number of training epochs, learning rate, and sequence length. Therefore, it is crucial to carefully tune these hyperparameters to achieve optimal performance and avoid overfitting.

References

Cirstea Razvan-Gabriel [et al.] Triformer: Triangular, Variable-Specific Attentions for Long Sequence Multivariate Time Series Forecasting--Full Version [Journal] // arXiv preprint arXiv. - 2022. - 2204.13767.

Feng Xiaoyu [et al.] SEFormer: Structure Embedding Transformer for 3D Object Detection [Journal] // arXiv preprint arXiv. - 2022. - 2209.01745.

Huber Peter J. Robust Estimation of a Location Parameter [Book Section] // Breakthroughs in Statistics: Methodology and Distribution / book auth. Kotz Samuel and Johnson Norman L.. - New York : Springer New York, NY, 1992.

Luo Liangchen [et al.] Adaptive Gradient Methods with Dynamic Bound of Learning Rate [Journal] // arXiv preprint arXiv. - 2019. - 1902.09843.

Zeng Pengyu [et al.] Muformer: A long sequence time-series forecasting model based on modified multi-head attention [Journal] // Knowledge-Based Systems. - October 27, 2022. - 109584 : Vol. 254.