Apply Transformer and Self-Supervised Learning to Multivariate Time Series Regression and Clasification

MDS6224 Group 22 Project Report

Luo Xinyang

118020047

May 14, 2022

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

1 Introduction

Multivariate time series(MTS) is an important kind of data in many areas. As the name suggests, MTS is a kind of data composed of serval time series data. In the financial area, we can view all the listed stock prices in the market as an MTS. By accurately predicting the price and form portfolio, we can make lots of profit. In the environment area, we can predict the rainfall and air quality index(AQI). In the medical area, we can view Electroencephalogram(EEG) and Electrocardiogram(ECG) data as MTS and use the classification result to help diagnose patients. Transformer [1] and its variants like vision transformer(ViT) [2] have achieved state-of-the-art performance in natural language processing(NLP) and computer vision(CV). Transformer and its encoder-decoder architecture have shown great success when dealing with sequential data. It is natural to apply it to the MTS problem. Self-supervised learning consists of unsupervised pretraining, and supervised finetuning is an outstanding technique often used together with the transformer in recent years. In this work, we apply self-supervised learning and transformer to the MTS problems. Also, this work presents a dual transformer architecture that enhances the locality of the transformer. All the Models above are tested on several public datasets and compared to some open benchmark methods. Our models show satisfying results in most of the tasks.

2 Related Work

2.1 Time Series Regression and Classification

There are many models for time series regression and classification. Classical method like dynamic time wrapping(DTW) [3] together with k-nearest-neighbor (KNN) [4] is successful in many traditional univariate time series problems. With the development of deep learning, recurrent neural network(RNN)-based models, which are designed for sequential data, are applied to the MTS problem. Among the variants of RNN, long-short-term-memory(LSTM) [5] is the most famous one. Many convolution neural networks (CNN) proved successful in computer vision are applied to the MTS problem. Among them, fully convolutional network(FCN) [6] and residual network(ResNet) [7] are widely used. In this work [8], the author proposed a baseline for multilayer perceptron(MLP), FCN, and ResNet on univariate time series regression. ROCKET [9] is extremely fast while maintaining outstanding performance.

2.2 Transformer

Transformer [1] is a successful architecture that achieves state-of-the-art performance in NLP and computer CV. The transformer encoder and decoder are composed of N stacked layers. Each layer contains two sublayers: a multi-head self-attention

and a feed-forward network. Each sublayer is followed by residual connection and Layernorm. That is $output = LayerNorm(x + Sublayer(x))$

multi-head self-attention have the following mechanism with $Q = K = V$.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_n)W^O$$

$$where\ head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

And Position-wise Feed-Forward Network consists of two linear layers and a ReLU activation function.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

A recent paper [10] shows that transformer outperforms traditional models like ARIMA, LSTM, Sequence to Sequence(Seq2Seq) + attention in univariate time series forecasting.

2.3 Self-supervised Learning

Unsupervised pretraining and supervised finetuning have gained lots of success in recent years. In NLP, BERT [11] masks 15% of WordPiece with a [mask] token and

uses a transformer encoder to recover the masked WordPiece. Another pretraining task of BERT is to judge whether sentence B is the next sentence following sentence A, known as next sentence prediction(NSP). After the pretraining, BERT uses the transformer encoder followed by some layers and supervised learning to fit the downstream task. MAE [12] is another state-of-the-art architecture in the CV area. MAE masks a large proportion(75%) of an image and then uses a ViT [2] encoder-decoder to recover the whole picture. After pretraining, MAE abandons the decoder and only uses the Vit encoder output with some following layer for finetuning.

For time series self-supervised learning, a recent paper [13] proposed a method. Divide the origin time series into lots of subseries, then use the subseries to predict the next value. Another approach [14] is to mask a proportion r of the input MTS. Each mask's length is controlled by a geometry distribution. Then use a transformer encoder to recover the origin MTS.

3 Methodology

3.1 Problem Description

We define a multivariate time series $X \in \mathbb{R}^{p \times T}$ as following,

$$X = \{X_1, X_2, ..., X_T\}$$

$$where\ X_t = \{x_t^1, x_t^2, ..., x_t^p\} \quad 1 \leq t \leq T$$

The classification task is given $X \in \mathbb{R}^{p \times T}$, find the category C that X belongs to. And the regression task is given $X \in \mathbb{R}^{p \times T}$, predict all the features in next time step X_{T+1} or predict X_{T+1}^i where $1 \leq i \leq p$ a certain feature in next time step.

3.2 Self-Supervised Model

Inspired by the work of MAE, we generate a binary mask $M \in \mathbb{R}^{p \times T}$ the same shape as input X. Each entry in M has a probability p of being 0 and otherwise 1. p is usually set as 40% to 75%. We use $\hat{X} = X \otimes M$ as input, then use a transformer encoder-decoder to get the recover $\tilde{X}$ and minimize the mean squared error(MSE). To let the model focus more on the recovering task, the masked and unmasked part contributes differently to the loss. We define the weighted MSE loss as follows, λ is

set to 0.5 by default. For datasets with missing values, we replace the missing value with 0, and the loss of the missing entry won't count for the total loss.

$$\mathcal{L}_{WeightedMSE}(X, \tilde{X}, \lambda) = \mathcal{L}_{MSE}(X \otimes M, \tilde{X} \otimes M) + \lambda \mathcal{L}_{MSE}(X \otimes \tilde{M}, \tilde{X} \otimes \tilde{M})$$

$$where\ \tilde{M} = 1 - M$$

$$\mathcal{L}_{MSE}(X, \tilde{X}) = \frac{1}{p \times T} \sum_{i=1}^{p} \sum_{t=1}^{T} (\tilde{x}_t^i - x_t^i)^2$$

Unlike BERT, which generates the static mask in preprocessing, the mask in our model is generated in each training step like roBERTa [15]. The dynamic mask itself is a data augmentation and can help learn the intrinsic pattern and also prevent the model from overfitting. Note that since we don't have the one-hot word vector as NLP, we replace the embedding layer with a linear layer with $W \in \mathbb{R}^{p \times h}$ that maps the feature to hidden dimension h. The positional encoding is set to learnable encoding as BERT, and the activation functions in the transformer block are Gaussian error linear units(GELU) [16]. For finetuning, we abandon the decoder, then put to encoder output to a 1d-convolution layer To reduce the time dimension. The convolution kernel has $in\_channels = T$, $out\_channels = 1$, and $kernel\_size = 1$. Following the convolution layer are a few dense layers that generate the final output

for downstream tasks.

3.3 Dual Transformer

Traditional transformers use global attention, which is good at finding global patterns and hierarchy. But patterns in time series involve lots of local patterns. Inspired by DSANet [17], we proposed the dual transformer architecture. One transformer encoder takes the original input to capture the global pattern. For the other part, we apply 1d-convolution and maxpooling to input X with respect to time dimension serval times, then send it into another transformer encoder. After that, we sum the outputs from two transformer encoders.

We apply casual convolution [18] here. For a casual convolution with kernel size k, we pad k-1 items only before time step 1. This ensures that the shape of X won't change, and the current time step won't contain future information. Also, each in-channel will only correspond to one out-channel. This is to avoid interference from different features and let the kernel focus on finding the local patterns in each time series. The maxpooling is modified in the same manner. The kernel size is set based on the time length of MTS.

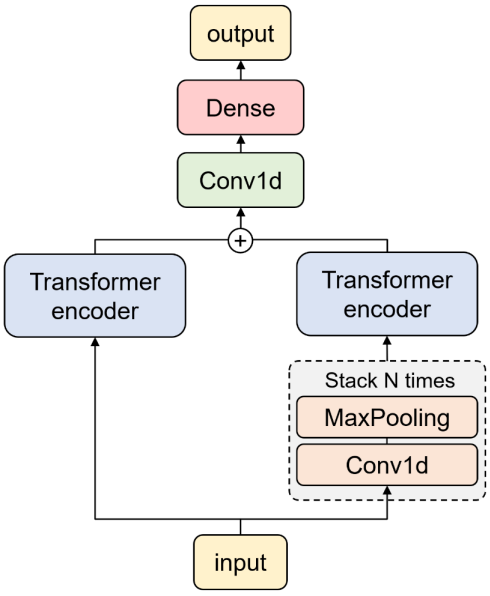

Figure 1: Dual Transformer Architecture

4 Experiments

In the experiment, we use five datasets from the time series extrinsic regression archive(TSER) [19] for regression and five datasets from the UEA multivariate time series classification archive [20] for classification. We keep the origin train test split so that we can compare to the public benchmark [14, 20, 21] more accurately and fairly. We don't use the widely used UCR time series archive [22] since it is all univariate. Since we didn't find any reported RNN-based model on the UEA time series archives or TSER archive. We implement a stacked LSTM for comparison. To ensure that LSTM won't suffer losses from the lack of parameters, we stacked layers are set to

five times the number of encoder blocks in the transformer.

We compare our model with LSTM, XGBoost [23], ROCKET, ResNet, and 5-NN-DTW in regression task and LSTM, XGBoost, ROCKET, and DTW in the classification task. Our model is trained with the Adam optimizer [24]. For regression, we evaluate the performance in terms of Rooted Mean Squared Error(RMSE), and for classification, we evaluate the performance in terms of accuracy rate.

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(y - \hat{y})^2}$$

Also, we implement a supervised only transformer with only the encoder, 1d-convolution, and dense output layer, which is the finetune architecture of the self-supervised model. This model serves as an ablation test to verify the effectiveness of the pre-training and dual structure.

5 Results

We use P-TT to denote pretrained time transformer, S-TT to denote supervised only time transformer, and Dual-TT to denote dual time transformer.

5.1 Regression

Table 1 shows the results of MTS regression task in terms of RMSE. The pretrained model and dual transformer are the second and third best models, better than the other two neural network models: ROCKET and ResNet. The pretrained model shows significant improvement (5-10%) in 4 datasets compared to the supervised only model. And dual transformer outperforms the single transformer in all five datasets with a 5% improvement compared to the pretrained version. All three variants of transformer perform (in terms of rank) worst on LiveFuelMoistureContent dataset, where only seven features are provided. All three variants perform best on BeijingPM25Quality and BeijingPM10Quality datasets. These two dataset contains missing value. Thus the lead may be explained by the better robustness and generalization ability of the transformer.

Dataset	P-TT	S-TT	Dual-TT	LSTM	XGBoost	ROCKET	ResNet	5-NN-DTW
BeijingPM10Quality	92.59	96.32	**89.44**	136.4	93.14	120.1	95.49	115.5
BeijingPM25Quality	61.99	62.28	**57.70**	112.9	59.50	62.77	64.46	72.72
AppliancesEnergy	3.218	3.532	3.697	3.815	3.49	**2.30**	3.07	4.02
BenzeneConcentration	0.838	0.983	0.781	8.016	**0.64**	3.36	4.06	4.87
LiveFuelMoistureContent	49.61	52.11	45.36	41.13	32.44	**29.41**	30.35	35.19
Average Rank	3.6	5.2	3.2	7.2	2.6	3.8	4	6.4

Table 1: Performance on MTS regression task, in terms of RMSE. Bold indicates best values.

5.2 Classification

Table 2 shows the result of MTS classification task in terms of accuracy rate. In this task, the dual transformer model and ROCKET tied for the first place with an average rank of 2.4. We can observe that the pretrained model shows no advantage compared to the supervised only model. Except for one dataset: Handwriting, where we only have 150 train cases but 26 classes. Still, the dual Transformer shows an obvious lead in three datasets and close performance in the remaining two datasets compared to the single transformer.

Dataset	P-TT	S-TT	Dual-TT	LSTM	XGBoost	ROCKET	DTW
FaceDetection	0.6498	0.6517	**0.6842**	0.5834	0.633	0.647	0.529
Heartbeat	0.7366	0.7317	0.7476	0.722	0.732	**0.756**	0.717
Handwriting	0.2118	0.1647	0.2094	0.1438	0.157	0.588	0.286
PEMS-SF	0.8266	0.8382	0.8902	0.7052	**0.983**	0.751	0.711
UWaveGestureLibrary	0.8133	0.8125	0.8531	0.6	0.759	**0.944**	0.903
Average Rank	3.4	4	2.4	6.6	4.4	2.4	4.8

Table 2: Performance on MTS classification task, in terms of accuracy rate. Bold indicates best values.

6 Analysis

In the experiment result above, we find some interesting phenomena. Pretraining shows significant effectiveness in most regression tasks but makes no almost difference in most of the classification tasks. This may be explained by the different properties of regression and classification MTS. Most regression MTSs are consecutive, and they

can generate the next item eternally like the BeijingPM10Quality dataset. As the day goes by, the MTS will keep generating. But classification MTSs are different. They are more like a record of a certain event with a clear beginning and end. Take the FaceDetection dataset as an example, X is EEG data of a subject when seeing a picture, and the task is to predict whether there is a human face in the picture or not. Thus the pretraining task is of predict the value is closer to the regression task. But predicting next the value itself makes less sense to MTS classification tasks.

For extreme small datasets, pretraining show great effectiveness. For Handwriting datasets with only 150 train cases (about 5 cases per label) and 850 test cases. Pretraining achieve a 5% improvement in accuracy rate. And for AppliancesEnergy, with only 96 train cases and 24 test cases, pretraining lower the RMSE by around 10%. This may give credit to the powerful data augmentation of dynamic masking in pretraining.

It is also worth noticing that on datasets with very small dimensions (3-7) like Handwriting, UWaveGestureLibrary, and LiveFuelMoistureContent transformer-based models are outperformed by light models like XGBoost and ROCKET. We believe that the huge total amount of parameters make the model hard to train and highly likely to overfit the train data.

The Dual transformer's success proves the lack of local pattern detection ability of

the origin transformer. The casual convolution and maxpooling to the input data can mitigate the problem.

Still, we have to admit that MTS data are highly noisy and information in it is sparse. And the importance of different features or different time period varies a lot. For some dataset, among all the features, only one or two feature contains essential information for the prediction task. Or only a small fraction of time determines the outcome. Unlike the NLP task, each world piece has very dense information, and the language model itself has an abundant global hierarchy structure. Thus the performance of the transformer on the MTS tasks is not quite outstanding as it is in NLP or CV.

Traditional RNN-based models like LTSM don't fit the MTS tasks. LSTM performs worst in almost all datasets. And its training process is also slow due to its serial structure.

7 Future Work

Considering the unsatisfying performance of the pretrained time transformer on MTS classification, we will try to modify the pretraining task for the classification dataset. We will come up with a task like the NSP in BERT. Divide a time series into two parts, and let the model judge whether time series B follows time series A or not.

It is worthwhile to combine convolution and transformer. We will try adding different kinds of convolution in different locations of the transformer. Other approachs that can enhance the locality should also be tried. It is also attractive to build a pretrained model with the dual transformer to see whether we can further improve the performance. This is not contained in this work simply due to the limited computational power, especially GPU memory.

The performance in the experiment means it can be applied to more real-world tasks. We hope to develop a commodity future pair trading strategy based on it. By inserting domain knowledge, feature engineering, and proper preprocessing, the model will present a better performance.

The framework of pretraining and finetuning can also apply to other areas like time series clustering and dimension reduction.

8 Conclusion

In this work, we test the transformer and self-supervised learning on multivariate time series regression and classification. The method includes an unsupervised pretraining that uses a transformer encoder-decoder to recover the masked input and a finetuning that uses the output of the encoder with some following net for downstream tasks. The performance benefits from pretraining on MTS regression tasks a lot. But

further modification and improvement are needed for MTS classification problem. We propose a dual transformer architecture that enhances locality as well. The result is satisfying .Both pretrained model and dual transformer are better than most traditional methods, and dual transformer beats other models in many datasets.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[3] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, USA:, 1994.

[4] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585. IEEE, 2017.

[9] Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

[10] Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317*, 2020.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

[13] Shayan Jawed, Josif Grabocka, and Lars Schmidt-Thieme. Self-supervised learning for semi-supervised time series classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 499–511. Springer, 2020.

[14] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, 2021.

[15] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[16] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[17] Siteng Huang, Donglin Wang, Xuehan Wu, and Ao Tang. Dsanet: Dual self-attention network for multivariate time series forecasting. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 2129–2132, 2019.

[18] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems*, 32, 2019.

[19] Chang Wei Tan, Christoph Bergmeir, Francois Petitjean, and Geoffrey I Webb. Monash university, uea, ucr time series extrinsic regression archive. *arXiv preprint arXiv:2006.10996*, 2020.

[20] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. The uea multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075*, 2018.

[21] Chang Wei Tan, Christoph Bergmeir, François Petitjean, and Geoffrey I Webb. Time series extrinsic regression. *Data Mining and Knowledge Discovery*, 35(3):1032–1060, 2021.

[22] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

[23] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.