Time Series Forecasting of Air Pollutant PM2.5 Using Transformer Architecture

K. Azhahudurai[1], Dr. V. Veeramanikandan[2]

[1]Department of Computer and Information Science, Annamalai University, Annamalai Nagar.
Email: *jkmaz477[at]gmail.com*

[2]Department of Computer Science, ThiruKolanjiappar Government Arts College, Vridhachalam.
Email: *klmvmani[at]gmail.com*

Abstract: ***Transformer architectures are widely used, especially in computer vision and natural language processing. Transformers have been used recently in a number of time-series analysis applications. An overview of the Transformer architecture and its uses in time-series analysis is given in the literature review. To improve performance, the Transformer's primary parts—the encoder/decoder, multi-head, positional encoding, and self-attention mechanism—have been updated. To implement time-series analysis, a few improvements to the original transformer architecture were adopted. Additionally, the optimal hyperparameters values for overcoming the difficulty of successfully training Transformers for time-series analysis are provided in this work. The effectiveness of the Transformer model in forecasting PM2.5 concentrations is examined in this paper. The dataset is pre-processed as a first step. In order to minimize the input parameters while taking into account their statistical significance, multi-collinearity among the independent variables is found using a Variance Inflation Factor (VIF). The proposed model have been trained to forecast PM2.5 concentrations up to one day ahead of time.***

Keywords: transformer architecture, time series analysis, self attention, hyperparameters, forecasting PM 2.5, multi-collinearity, Variance Inflation Factor

1. Introduction

Transformers are a subclass of machine learning models that primarily learn via the scaled dot-product operation or self-attention. One of the most difficult natural language processing (NLP) tasks, neural machine translation, is where transformers were first proposed [1]. Transformers have recently shown effective in solving a range of machine learning issues and achieving cutting edge performance [2]. Examples from other fields, besides classical NLP tasks, are image classification [3], language and image generation [4], object detection and segmentation [5], sequential decision-making in reinforcement learning [6], multi-modal (text, speech, and image) data processing [7], and tabular and time-series data analysis [8].

Transformer-based time-series analysis is the main topic of this research. Sequentially recorded samples, observations, or features over a period of time are referred to as time-series data. In many real-world applications, where data is recorded over a predetermined sampling interval, time-series datasets are a common occurrence. Stock prices, digital voice signals, traffic measures, weather pattern sensor data, biological measurements, and numerous demographic data types collected over time are a few examples. Time-series analysis can involve handling numerical data for a variety of purposes, such as classification, forecasting, and prediction. Using a variety of models, including autoregressive (AR), moving average (MA), auto-regressive moving average (ARMA), AR Integrated MA (ARIMA), and spectral analysis techniques, are part of statistical methodologies.99% of people on Earth, according to estimates from the World Health Organization, reside in areas with air pollution levels higher than recommended. It is also mentioned that 6.7 million deaths annually are attributed to exposure to ambient air pollution. These pollutants have an impact on the quality of the air, particularly ambient fine particle air pollution, or PM2.5. Among other things, burning agricultural waste, industrial processes, and automobiles all release this kind of pollution. When exposure to high PM2.5 concentrations persists for several years, they may become hazardous.

The current work implements transformer architecture for PM2.5 concentration estimation in ambient air based on the Transformer model. To enhance the outcomes, the suggested architecture learns the spatiotemporal trends of air pollutants, including meteorological data. The remainder of the paper is organized as follows. A thorough analysis of the relevant literature was provided in Section 2. Section 3 presented the architecture of the transformer used for time series forecasting. Section 4 reports the experimental results along with the dataset, study metrics, and a discussion of the outcomes. Lastly, Section 5 provides conclusions.

2. Related Work

The research community has widely adopted various suggested machine learning models with specialized parts and architectures for managing the sequential nature of data. Recurrent Neural Networks (RNNs) and their well-liked variations, such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) [9], [10], [11], [46] and [12], are the most well-known of these machine learning models. These models use the well-known gradient descent approach to optimize unknown model parameters while processing batches of data sequentially, one sample at a time. Backpropagation through time (BPTT) is used to calculate the gradient information needed to update the model parameters [13]. Numerous applications have seen the successful employment of LSTMs and GRUs [14], [15], [16], [17], [18], and [45]. However, they have a number of drawbacks as a result of the BPTT difficulties and the sequential processing of incoming data, particularly when handling datasets with lengthy relationships. Moreover, bursting and disappearing gradient issues plague the training of LSTM and GRU models [19], [20]. The gradient descent algorithm (using BPTT) may not be able to update the model parameters when processing lengthy sequences because the gradient information is lost.

Furthermore, the parallel computations provided by graphical processing units (GPUs), tensor processing units (TPUs), and other hardware accelerators are typically not advantageous for these models [21]. To some extent, gradient-related issues may be mitigated by LSTMs and GRUs with the aid of specific architectural adjustments and training techniques. However, the efficacy and efficiency of RNN-based models are impacted by difficulties in learning from lengthy data sequences with the restricted use of parallelization provided by contemporary hardware [22]. Sequential data can be computed in parallel using the Transformer architecture [23] without significantly increasing network complexity [24]. Transformers can leverage the parallel processing power of Tensor Processing Units (TPUs) and Graphics Processing Units (GPUs) due to their architecture [25]. Transformers, as opposed to RNNs and their variants, do not experience vanishing gradients because of the attention-based operations, which allow them to correlate information from all elements in the sequence to each other in parallel [26], [27], [28], [43], [44].

Long-term and multi-variate time-series forecasting has been significantly enhanced by transformers [29], [30]. Long sequence modeling is hampered by the self-attention mechanism's high memory and computational complexity requirements. To maximize Transformer performance for timeseries tasks, a number of changes have been suggested in the literature [31], [32], and [33]. It can be difficult to train large Transformer models, particularly for large datasets. In the literature, numerous methods for effectively training massive Transformer models have been suggested. Layer-wise adaptive large batch optimization [34], progressive training [37], distributed training [35], knowledge inheritance [36], and mapping smaller models' parameters to larger models' initialization [38] are some of these methods.

3. Overview of Time Series Transformer Architecture

The Transformer, a sequence-to-sequence model with an encoder-decoder configuration, accepts a sequence of words from the source language, which produces the translation in the target language [1]. The model must learn to encode the source sequence into a fixed-length representation that it can then decode to produce the target sequence in an auto-regressive manner because the two sequences' lengths and vocabulary sizes aren't always the same [42]. One limitation of this auto-regressive feature is that, in order for the translated sequences to be generated, information must propagate back to the beginning of the sequence. For time-series analysis, the same restriction applies. How far back the influence of a particular data sample can be taken into account during learning has limited the capabilities of machine learning models. Sometimes, instead of the training examples being generalized to new data, the auto-regressive nature of machine learning model training results in the memorization of previous observations.

Transformers solve these problems by jointly attending to and encoding the ordered information during the analysis of current data instances in the sequence using positional encoding and self-attention techniques. These methods do away with the traditional idea of recurrence while maintaining the sequential information necessary for learning. Transformers can now take advantage of the parallelism provided by GPUs and TPUs thanks to these techniques. There have been some recent studies attempting to integrate recurrent components into Transformer designs.

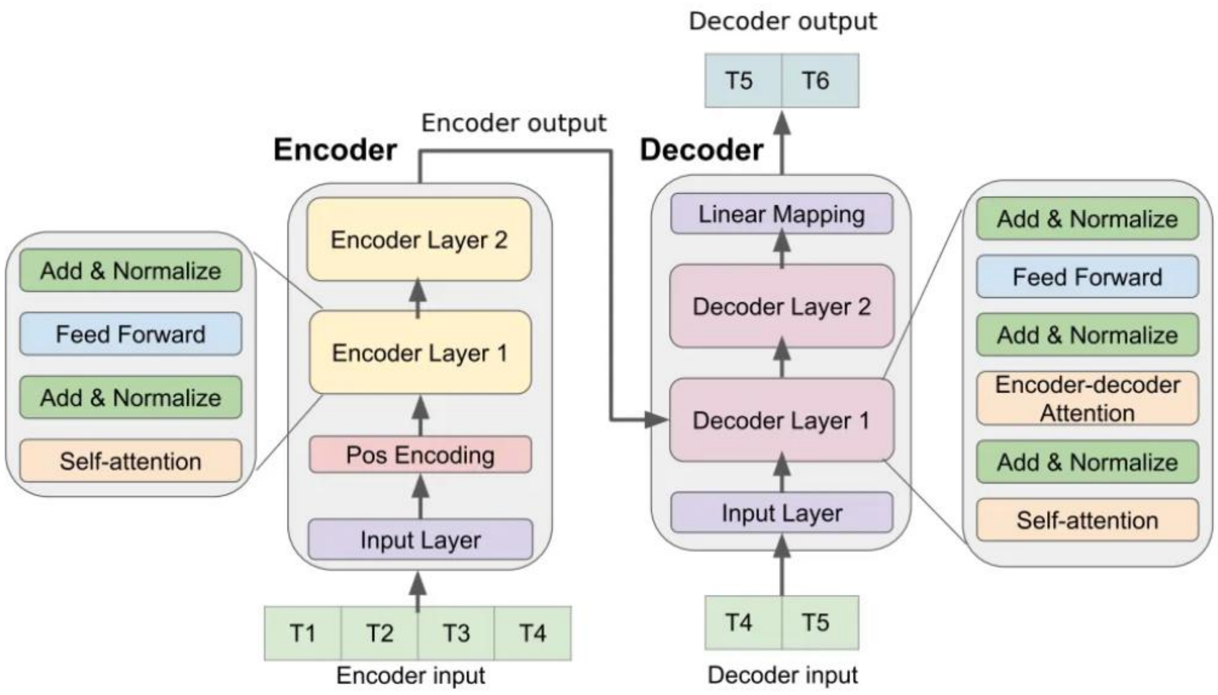

Figure 1: Schematic view of Transformer based forecasting [39]

The foundation of the Transformer architecture (shown in Fig. 1) is the use of the dot product to identify associations or relationships among different input segments. The weighted dot product of these input vectors, xi, with one another is the self-attention operation. There are two steps in the self-attention operation. A normalized dot product between each pair of input vectors in an input sequence is calculated in the first step. The softmax operator, which scales a given set of numbers so that the output numbers add up to unity, is used to perform the normalization. The normalized correlations between an input segment (xi) and every other segment (j = 1,..., n) was calculated using below mathematical expression:

$$w_{ij} = softmax\left(x_i^T x_j\right) = \frac{e^{x_i^T x_j}}{\sum_k e^{x_i^T x_k}} \qquad \text{Eq. 1}$$

where $\sum_{j=1}^{n} w_{ij} = 1$ and $1 \leq i, j \leq n$. The second step is to find a new representation z_i, which is a weighted sum of all the input segments $\{x_i\}_{j=1}^{n}$, for a given input segment x_i.

$$z_i = \sum_{j=1}^{n} w_{ij} x_j, \forall 1 \leq i \leq n \qquad \text{Eq. 2}$$

The sum of the weights w_{ij} for every input segment x_i is 1. As a result, the input vector x_j with the largest attention weight w_{ij} and the output vector z_i will be comparable. The highest correlation value, as determined by the normalized dot product between x_i and x_j, has consequently led to the largest attention weight. Multiple levels of correlation information may be present in the input data X, and processing the data in various ways may aid in the learning process. Multiple self-attention heads that work in parallel on the same input, extracting different levels of correlation between the input data using different weight matrices W_q, W_k, and $W_v$was used.
Multi-head self-attention involves three operations that can be summed up in three steps.
Step 1 - Creating Several Sets of Unique Query, Key, and Value Vectors
Step 2 - Parallel implementation of scaled dot products
Step 3 - Concatenating and linearly combining the outputs

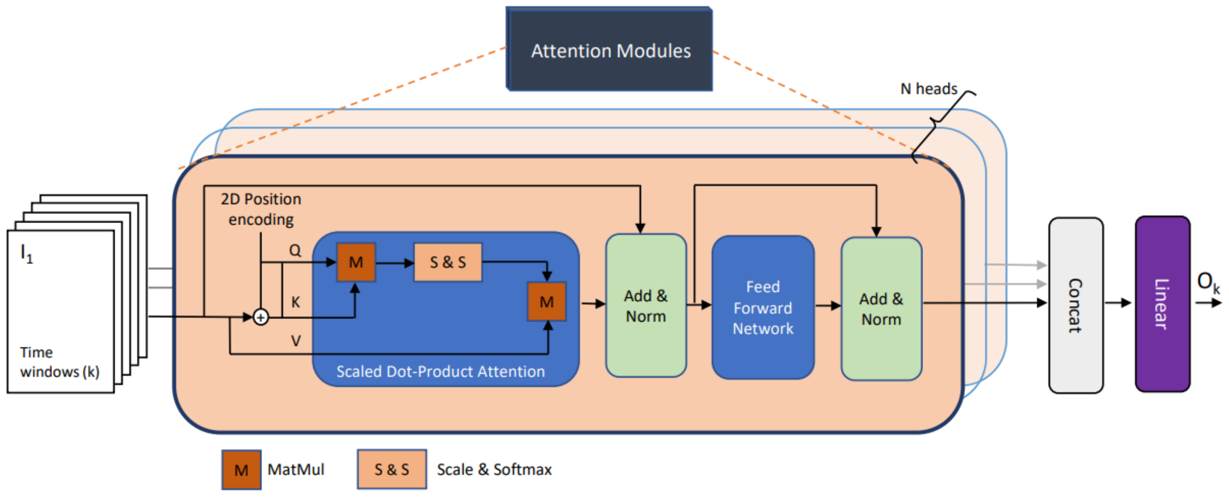

Figure 2: Schematic view of Transformer's Encoder with Attention modules [41]

The encoder and decoder component types are the two main component types that make up the Transformer architecture as presented in Fig. 2. A feed-forward layer and a multi-head self-attention layer coupled back-to-back with residual connections and normalization layers make up an encoder block. Deep neural networks are frequently trained using residual connections, which aid in learning and stabilization. Neural networks also frequently use the layer normalization operation to process sequential data. It facilitates the model training's quicker convergence. Two linear layers with a ReLU activation function make up the feed-forward layer. An encoder block's output feeds into the subsequent encoder block. Similar layers and operations make up each encoder and decoder block. Nevertheless, a decoder gets two inputs: one from the encoder that came before it and another from the one that came after. Three layers are present in a decoder: feed-forward layer, encoder-decoder attention layer, and multi-head self-attention layer. Remaining connections and layer normalization processes exist. The output of the final encoder is used to create a set of key and value vectors inside the encoder-decoder attention layer. The multi-head self-attention layer's output, which comes before the encoder-decoder layer, is used to create the query vectors. Depending on the problem being solved, a Transformer model may have stacks of multiple encoder and decoder blocks. The multiple hidden layers found in conventional neural networks are reminiscent of the stacked encoder/decoder blocks. It's crucial to remember that, typically, neither the encoder nor the decoder's processing results in a reduction of the representation's dimensions. The word sequence that has been mapped into word embeddings and has PEs added is the input for the first encoder block.

4. Experiment & Results

The Transformer architecture used a variety of techniques to stabilize the gradients while deeper networks were being trained. A deeper network can be trained by utilizing residual connections. Subsequently, an adaptive optimizer (Adam) was combined with layer normalization operations to offer various learning rates for various parameters. Transformers are sensitive to the learning rate, just like the majority of other deep learning models. Transformers can converge more quickly than conventional sequence models, provided that they have an ideal learning rate. A decline in performance is often seen in the first few epochs. Nonetheless, the model will typically begin to converge to better values after a few epochs. We employed a warm-up learning rate strategy that rises linearly for the first N training steps and falls proportionately to the step number's inverse square root, $1/\sqrt{N}$.

4.1 Dataset

The experiments were performed on the dataset containing the PM2.5 of the five Chinese cities including the meteorological data for each of the city over the duration Jan 01, 2010 and Dec 31, 2015. Dew point, temperature, humidity, atmospheric pressure, cumulated precipitation, and wind speed were used for predicting the value of the PM2.5 [42].

Table 1: Dataset Description

Item	Description
Characteristics	Time Series and Multivariate
Number of samples	43824
Number of Attributes	06 (weather and atmospheric parameters)
Period of Collection	2010 – 2015
Frequency	Hourly Data

4.2 Data Preprocessing

A statistical phenomenon known as multicollinearity happens when there is a strong correlation between two or more independent variables in a regression model. Stated differently, multicollinearity denotes a robust linear correlation between the predictor variables. Regression analysis may become challenging as a result, as it becomes more challenging to precisely identify the contributions of each independent variable to the dependent variable. It can be more difficult to interpret the data and derive significant inferences from the model when multicollinearity results in unstable and unreliable coefficient estimates. Regression models must be validated and robustened, and multicollinearity must be identified and addressed. When two or more independent variables in a data frame in a regression model have a high correlation with one another, this is known as multicollinearity.

The estimated regression coefficients may grow large and unpredictable when multicollinearity is present, making it difficult to draw valid conclusions about how the predictor variables affect the response variable. It's possible that multicollinearity has less of an impact on the machine-learning model's accuracy. However, we risk losing our ability to reliably assess the contributions of each individual feature in the model, which could pose an interpretability issue. A dataset can be examined for multicollinearity in a number of ways, including by computing the correlation matrix between the independent variables and utilizing the Variance Inflation Factor (VIF). Regularization and feature selection are two methods that can be used to choose a subset of independent variables that do not have a strong correlation with one another in order to address multicollinearity. The most popular approach, known as variance inflation factors, or VIFs, was applied in this work.

The correlation strength between the independent variables is ascertained by VIF. Regressing a variable against all other variables yields the prediction. To find out how well an independent variable is described by the other independent variables, one can calculate its R^2 value. A high R^2 value indicates a strong correlation between the variable and the other variables. This is represented by the VIF, which is as follows:

$$VIF = \frac{1}{1 - R^2} \qquad \text{Eq. 3}$$

Table 2: Results of VIF Analysis

Variable	VIF
DewPoint	**3.75**
Temp	**5.18**
Combined wind direction	**2.63**
Pressure	**4.24**
Cumulated wind speed	**1.56**

In the experiments for training the model only four out of five available parameters were used. Temperature was removed as VIF value was greater than 5. The threshold value was fixed based on the literature study conducted.

4.3 Proposed Transformer Architecture Improvements

Several improvements to the Transformer architecture have been put forth in an effort to address some of the issues surrounding the attainment of stable training with deeper architectures. Improved weight initialization methods and the relocation of layer normalization operations have been the main optimizations done to balance residual dependencies. More stable training has resulted from these enhancements, and in certain situations, they have even removed the need to employ some of the original architecture's suggested practices. Post-layer normalization, or post-LN, is the term used to describe the original Transformer architecture, in which the layer normalization is situated outside the residual block. Post-LN requires a learning rate warm-up strategy and converges much more slowly. Using a pre-layer normalization (pre-LN) Transformer, as suggested in the literature, this problem was solved and it was demonstrated that gradients could converge more quickly with no warm-up. This can be accomplished by balancing the residual dependencies and managing the gradient magnitudes with the Pre-LN Transformer [40]. Despite not requiring learning rate warm-up, the Pre-LN Transformer architecture performs less well empirically than the Post-LN Transformer architecture (Fig. 3).

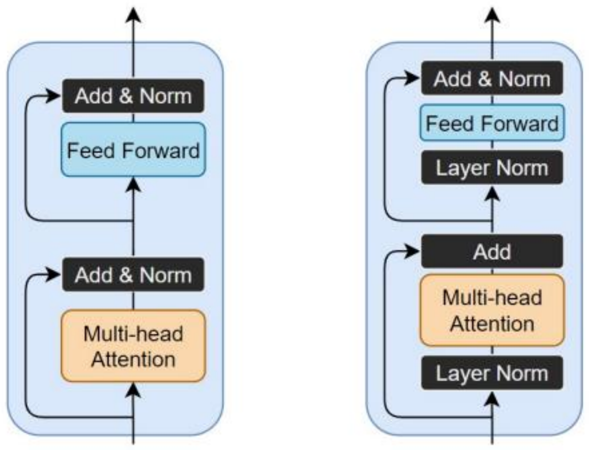

Figure 3: Schematic view of post-LN Transformer and pre-LN Transformer [40]

It is difficult to train these deep Transformer models from scratch on small datasets because they require large datasets. For small datasets to function effectively, pre-trained models and small batch sizes are needed. However, training more Transformer layers becomes more challenging due to these two requirements.

The updates' variance increases when a small batch size is used. Generally speaking, the model may not generalize well, even with large batch sizes. Improved initialization strategies may occasionally optimize the model and improve its performance on smaller datasets. For Transformers, the most popular scheme is Xavier initialization. The goal of the experiments was to determine how learning rate affected model performance and how those effects related to one another. The findings showed that while too high of a learning rate can result in non-convergence, small learning rates typically have slower convergence. During the training, gradient clipping was applied. If there are too many steps, this method usually prevents divergence and exploding gradients. Transformers avoid using other gradient clipping techniques that might cause slower convergence, such as selecting a step size based on the batch size.

With reference to Root Mean Square Error" (RMSE), the Transformer architecture outperformed sequence-to-sequence models with attention, LSTMs, and ARIMA. In the second experiment, multivariate time-series data were used to test the Transformer architecture. But the outcomes did not significantly improve as a result of this. In the third experiment, each scalar input x_t is embedded into a d-dimensional time-delay space to create Time-delay embedding (TDE).

This study takes into account a number of metrics, including Index of Agreement (IA), Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and Mean Absolute Error (MAE). In time series forecasting, these metrics are frequently employed as assessment metrics. The values obtained with the RMSE, MAPE, and MAE metrics close to 0 indicate a more accurate model. IA is a bounded, non-dimensional measure of the model prediction error degree, with values closer to 1 denoting a better match, according to [20]. The following mathematical formulas, which are listed in Table 3, were used to calculate these metrics:

Table 3: Performance Metrics

Metric	Formula
RMSE	$\frac{1}{n}\sum_{i=1}^{n}\sqrt{(pred(i)-gt(i))^2}$
MAPE	$\frac{1}{n}\sum_{i=1}^{n}\lvert pred(i)-gt(i)\rvert$
IA	$1-\frac{\sum_{i=1}^{n}(pred(i)-gt(i))^2}{\sum_{i=1}^{n}(\lvert pred(i)-\overline{gt}\rvert+\lvert gt(i)-\overline{gt}\rvert)^2}$

where$n$ is the total number of samples, $pred(i)$ is the model prediction for the i^{th} sample and $gt(i)$ is the respective ground truth. As was already mentioned, PM2.5 concentrations are predicted using a transformer architecture. Furthermore, two distinct architectures—the Long-Sort Term Memory RNN and the LSTM-RNN with attention—have also been taken into consideration. The variance inflation factors of the model were used to select the meteorological variables. The MinMaxScaler estimator is used to transform features for the pre-processing of data by scaling each feature to a specified range (e.g., between zero and one). The PM2.5 concentrations in ambient air input data were reframed as a supervised learning problem because they matched a multivariate time series.

Table 4: Comparison of proposed model’s performance

Models	RMSE	MAPE	MAE	IA
LSTM	6.19	0.72	4.38	0.80
LSTM + Attention	5.63	0.50	4.33	0.83
Transformer	4.98	0.48	3.86	0.85

For every model, the activation function, optimizer, learning rate, and batch size were fixed. The size of each Multi-Head Attention module used in transformer models and the number of attention heads it contains was fixed as values 8 and 10, respectively, in the proposed model. Similarly, Feed Forward Part, which corresponds to the transformer model's encoder, was set to 10. The results produced by the transformer model are superior to those produced by the MLP and LSTM models. Table 3 displays the obtained results for each of the suggested models. In the case of Transformer-based PM 2.5 estimation, the prediction results using models demonstrate a low error of RMSE and MAE metrics (4.98 and 3.86, respectively). Similarly, the obtained result for the IA metric—a standardized measure of the degree of model prediction error that varies between 0 and 1—was 0.83. Because the LSTM model is unable to learn large order dependence in sequence prediction problems, the highest error was obtained for time windows of 24 hours.

5. Conclusion

When it comes to handling time-series tasks, the Transformer architecture has shown to be an effective substitute for LSTMs and RNNs, while also surpassing their drawbacks. There have been suggestions for alterations to the original Transformer architecture in order to handle time-series data effectively. To train Transformer models effectively, a number of best practices have been implemented. This study's VIF correlation analysis highlights how crucial it is to eliminate meteorological variables that aren't necessary for the model training process. The accuracy of trained models is ascertained by comparing the three models. Results from experiments are presented along with comparisons between them, which demonstrate improvements over the obtained results when the transformer model is used. The experimental result demonstrates how the transformer models' attention modules can be used to identify pertinent features in the data and improve the accuracy of PM2.5 concentration estimates. The precision of PM2.5 concentrations that transformer models yielded could serve as a foundation for the creation of fresh ideas that make use of attention modules.Future research will address the open challenge of developing robust, self-aware Transformer architectures using uncertainty estimation in time-series prediction.

References

[1] AshishVaswani, Noam Shazeer, NikiParmar, JakobUszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and IlliaPolosukhin. Attention is all you need. Advances in Neural Information Processing Systems, 30, 2017.

[2] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and XipengQiu. A survey of transformers. AI Open, 3:111–132, 2022.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, XiaohuaZhai, Thomas Unterthiner, MostafaDehghani, 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.

[4] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao,

Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. arXiv preprint arXiv:2304.02643, 2023.

[5] Jiasen Lu, Christopher Clark, Rowan Zellers, RoozbehMottaghi, and AniruddhaKembhavi. Unified-IO: A Unified Model for Vision, Language, and Multi-Modal Tasks. arXiv preprint arXiv:2206.08916, 2022.

[6] Lili Chen, Kevin Lu, AravindRajeswaran, Kimin Lee, Aditya Grover, MishaLaskin, Pieter Abbeel, AravindSrinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. Advances in Neural Information Processing Systems, 34:15084–15097, 2021.

[7] AsimWaqas, AakashTripathi, Ravi P Ramachandran, Paul Stewart, and GhulamRasool. Multimodal Data Integration for Oncology in the Era of Deep Neural Networks: A Review. arXiv preprint arXiv:2303.06471, 2023.

[8] InkitPadhi, Yair Schiff, Igor Melnyk, MattiaRigotti, Youssef Mroueh, Pierre Dognin, Jerret Ross, Ravi Nair, and Erik Altman. Tabular transformers for modeling multivariate time series. In 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3565–3569, Toronto, 2021.IEEE.

[9] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning.arXiv preprint arXiv:1506.00019, 2015.

[10] SeppHochreiter and Jürgen Schmidhuber.Long Short-Term Memory. Neural Computation, 9(8):1735–1780, 11 1997.

[11] Junyoung Chung, CaglarGulcehre, KyungHyun Cho, and YoshuaBengio. Empirical evaluation of gated recurrent neural networks on sequence modeling.arXiv preprint arXiv:1412.3555, 2014.

[12] DimahDera, Sabeen Ahmed, Nidhal C. Bouaynaya, and GhulamRasool.Trustworthy uncertainty propagation for sequential time-series analysis in rnns. IEEE Transactions on Knowledge and Data Engineering, pages 1–13, 2023.

[13] Gang Chen. A gentle tutorial of recurrent neural network with error backpropagation.arXiv preprint arXiv:1610.02583, 2016.

[14] KE ArunKumar, Dinesh V Kalaga, Ch Mohan Sai Kumar, Masahiro Kawaji, and Timothy M Brenza.Forecasting of COVID-19 using deep layer recurrent neural networks (RNNs) with gated recurrent units (GRUs) and long short-term memory (LSTM) cells. Chaos, Solitons& Fractals, 146:110861, 2021.

[15] ZhengpingChe, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. Scientific reports, 8(1):6085, 2018.

[16] Luca Di Persio and OleksandrHonchar. Recurrent neural networks approach to the financial forecast of google assets. International journal of Mathematics and Computers in simulation, 11:7–13, 2017.

[17] ApekshaShewalkar, DeepikaNyavanandi, and Simone A Ludwig. Performance evaluation of deep neural networks applied to speech recognition: RNN, LSTM and GRU. Journal of Artificial Intelligence and Soft Computing Research, 9(4):235–245, 2019.

[18] Matthew Dixon and Justin London. Financial forecasting with α-rnns: A time series modeling approach. Frontiers in Applied Mathematics and Statistics, 6:551138, 2021.

[19] HongxiaoFei and Fengyun Tan. Bidirectional grid long short-term memory (bigridlstm): A method to address context-sensitivity and vanishing gradient. Algorithms, 11(11):172, 2018.

[20] António H. Ribeiro, KoenTiels, Luis A. Aguirre, and Thomas Schön. Beyond exploding and vanishing gradients: analysingrnn training using attractors and smoothness. In Silvia Chiappa and Roberto Calandra, editors, Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, volume 108 of Proceedings of Machine Learning Research, pages 2370–2380. PMLR, 26–28 Aug 2020.

[21] Julia El Zini, YaraRizk, and Mariette Awad.An optimized parallel implementation of non-iteratively trained recurrent neural networks. Journal of Artificial Intelligence and Soft Computing Research, 11(1):33–50, 2021.

[22] Alex Sherstinsky. Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network.Physica D: Nonlinear Phenomena, 404:132306, 2020.

[23] Yi Tay, MostafaDehghani, DaraBahri, and Donald Metzler. Efficient transformers: A survey. ACM Computing Surveys, 55(6):1–28, 2022.

[24] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. 2020.

[25] Norman Jouppi, Cliff Young, NishantPatil, and David Patterson.Motivation for and Evaluation of the First Tensor Processing Unit. IEEE Micro, 38(3):10–19, 2018.

[26] RazvanPascanu, Tomas Mikolov, and YoshuaBengio.On the difficulty of training recurrent neural networks. In SanjoyDasgupta and David McAllester, editors, Proceedings of the 30th International Conference on Machine Learning, volume 28 of Proceedings of Machine Learning Research, pages 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[27] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation.arXiv preprint arXiv:1906.01787, 2019.

[28] AnkurBapna, Mia Chen, OrhanFirat, Yuan Cao, and Yonghui Wu. Training deeper neural machine translation models with transparent attention. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3028–3033, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[29] Yunhao Zhang and Junchi Yan.Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In The Eleventh International Conference on Learning Representations, 2023.

[30] YuqiNie, Nam H Nguyen, PhanwadeeSinthong, and JayantKalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. arXiv preprint arXiv:2211.14730, 2022.

[31] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and SchahramDustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In International conference on learning representations, 2021.

[32] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In KamalikaChaudhuri, Stefanie Jegelka, Le Song, CsabaSzepesvari, Gang Niu, and Sivan Sabato, editors, Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pages 27268–27286. PMLR, 17–23 Jul 2022.

[33] Weiqi Chen, Wenwei Wang, BingqingPeng, Qingsong Wen, Tian Zhou, and Liang Sun. Learning to rotate: Quaternion transformer for complicated periodical time series forecasting. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD ’22, page 146–156, New York, NY, USA, 2022. Association for Computing Machinery.

[34] Yang You, Jing Li, SashankReddi, Jonathan Hseu, Sanjiv Kumar, SrinadhBhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. arXiv preprint arXiv:1904.00962, 2019.

[35] Yanping Huang, Youlong Cheng, AnkurBapna, OrhanFirat, Dehao Chen, Mia Chen, HyoukJoong Lee, JiquanNgiam, Quoc V Le, Yonghui Wu, and zhifeng Chen. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019.

[36] Yujia Qin, Yankai Lin, Jing Yi, Jiajie Zhang, Xu Han, Zhengyan Zhang, Yusheng Su, Zhiyuan Liu, Peng Li, Maosong Sun, et al. Knowledge inheritance for pre-trained language models. arXiv preprint arXiv:2105.13880, 2021.

[37] C. Li, B. Zhuang, G. Wang, X. Liang, X. Chang, and Y. Yang. Automated progressive learning for efficient training of vision transformers. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 12476–12486, Los Alamitos, CA, USA, jun 2022. IEEE Computer Society.

[38] Peihao Wang, Rameswar Panda, Lucas TorrobaHennigen, Philip Greengard, Leonid Karlinsky, RogerioFeris, David Daniel Cox, Zhangyang Wang, and Yoon Kim. Learning to grow pretrained models for efficient transformer training.arXiv preprint arXiv:2303.00980, 2023.

[39] Wu, Neo, et al. "Deep transformer models for time series forecasting: The influenza prevalence case." *arXiv preprint arXiv:2001.08317* (2020).

[40] Ahmed, Sabeen, et al. "Transformers in time-series analysis: A tutorial." *Circuits, Systems, and Signal Processing* 42.12 (2023): 7433-7466.

[41] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in Neural Information Processing Systems 30 (2017)

[42] Liang, X., Zou, T., Guo, B., Li, S., Zhang, H., Zhang, S., Huang, H. and Chen, S. X. (2015). Assessing Beijing's PM2.5 pollution: severity, weather impact, APEC and winter heating.Proceedings of the Royal Society A, 471, 20150257.

[43] V. Veeramanikandan and M. Jeyakarthic, "A Futuristic Framework for Financial Credit Score Prediction System using PSO based Feature Selection with Random Tree Data Classification Model," 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2019, pp. 826-831.

[44] Veeramanikandan, Varadharajan&Jeyakarthic, M..(2020). Parameter-Tuned Deep Learning Model for Credit Risk Assessment and Scoring Applications.Recent Advances in Computer Science and Communications. 13.

[45] Veeramanikandan, V., and M. Jeyakarthic. "An ensemble model of outlier detection with random tree data classification for financial credit scoring prediction system." International Journal of Recent Technology and Engineering (IJRTE) 8.3 (2019): 2277-3878.

[46] Veeramanikandan, V., and M. Jeyakarthic. "Forecasting of Commodity Future Index using a Hybrid Regression Model based on Support Vector Machine and Grey Wolf Optimization Algorithm." International Journal of Innovative Technology and Exploring Engineering (IJITEE) 10.10 (2019): 2278-3075.