Exploring transformer-based neural network architectures on time-series prediciton

Candidates: 50507, 41538, 56071

Abstract

For retail businesses, workforce optimisation is becoming more and more crucial since it lowers labour costs by predicting the appropriate amount of staff needed to work at a given moment using predictive analytics. This research project proposes a deep learning approach to staffing optimization using multivariate time-series prediction. Specifically, we compare the performance of an ARIMA model with the modern Transformer-based approach. Using historical data that has been trained on the number of visitors and the number of employees working at a given time we evaluate each performance on a test set. Our results show that the Transformer-based approach outperforms the ARIMA model in terms of prediction accuracy and computational efficiency. The findings suggest that the modern way is more efficient at predicting staffing resources in retail companies and potentially beneficial in reducing labor costs and ensuring customer satisfaction.

1. Introduction

Retail companies often face issues when it comes to optimizing their staffing resources, which can result in a significant expense if not optimized efficiently. Staffing optimization involves ensuring your environment that the right number of employees are working at a given time to meet customer demands. According to a report by McKinsey & Company the use of predictive analytics in workforce optimization can lead to a 3-5% increase in productivity and a 2-3% reduction in labor costs, highlighting the growing importance of workforce optimization (Amar et al., 2022). The data of daily staffing operation follows the time series. By leveraging historical data that is central to the operations of the business and make staffing prediction, such as the number of visitors or the number of employees working at a given time into a predictive forecasting series, company would generate more accurate forecasts and improve the utilisation of resources for higher revenue. Therefore, it is necessary for company to generate different types of model of time series prediction for higher accuracy. A traditional choice for multivariate time-series prediction is the auto-regressive Integrated Moving Average (ARIMA) model introduced in 1970 (Hochreiter & Schmidhuber, 1997). It makes an effort to predict the future values of the target variable using past observations of the variable (Mao et al., 2022). Nowadays, new deep learning architecture becomes more and more popular with its briiliant attention mechanism (Vaswani et al., 2017). It is worth noting that transformer would bring different perspective in time series prediction with more complex and dynamic data.

By contrasting a more conventional time series prediction model, ARIMA, with a more intricate neural network architecture, Transformer, this report investigates the use case of multivariate time-series prediction to improve staffing resources in a retail company. A deeper knowledge of the generating transformer model will be attained by contrasting the performance of various models with training and validation loss, and time series prediction in real-world applications will be tested in order to conduct a further in-depth study of deep learning field.

The following is a solution to the issue of staffing optimisation in retail businesses. The traditional time series prediction model ARIMA is used to provide the baseline for the forecasting after cleaning and standardising real-world data. Then, based on the fundamental property of the transformer's self-attention mechanism, a simplified version of the transformer is developed and compared to the ARIMA model. Additionally, a transformer with a more complex architectural design is created to analyse the effect on the performance of the prediction. One important aspect of time-series prediction is the number of previous time steps used as input variables, therefore two distinct values of "window size" are trained to see which one performs better.

There are three primary findings from the training described above:

1. Transformer-based deep learning models get less loss

. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

and better performance compared to conventional ARIMA model.

2. The more complex structure of transformer model with encoder-decoder mechanism achieves better performance compares to the simple transformer model with only self-attention mechanism.

3. With the increase of the window size, the complex transformer model increases the training and validation loss instead. In other word, the addition of window size doesn't bring better fit for the model due to the increased complexity of the model.

2. Related Work

This paper relates more broadly to the literature on time-series forecasting, and more specifically to papers applying transformer-based neural networks to time-series forecasting. A time-series is a collection of observations organised according to time. Time-series forecasting models project demand using mathematical methods based on historical data (Montero-Manso & Hyndman, 2021). Additionally, the paper compares forecasting using neural networks versus conventional statistically methods. One of the most crucial areas to concentrate on in the retail industry is optimisation, and in the past, the ARIMA model was the most often used algorithm in the industry. We shall investigate the use of transformer-based architecture and ARIMA models in a retail setting in this literature study.

2.1. ARIMA Model

The conventional literature on the time-series forecasting often relates to using Auto-Regressive Integrated Moving Average (ARIMA) forecasting algorithm as the traditional method to predict future values based on previous data. Based on previously observed data, the ARIMA model is regarded as stationary time-series data by combining its lagged time-series values (”AR”) and moving average of delayed predicted errors (”MA”) (Hochreiter & Schmidhuber, 1997). The performance of ARIMA versus neural network approaches is being actively compared as the gold standard for time-series forecasting. Scholars compare Recurrent Neural Network (RNN) and ARIMA models for predicting wholesale food prices and show that the accuracy of the combined RNN and CNN model is greater (Menculini et al., 2021). When other scholars evaluate the performance of ARIMA models and deep learning models for weather prediction, they find that deep learning models perform better (De Saa & Ranathunga, 2020). In this study, the performance of deep learning models will be compared against the benchmark time-series forecasting model, the ARIMA model.

In the retail industry, it is also thought to be effective at forecasting consumer demand, customer traffic, or inventory optimisation, which indirectly controls labor optimisation. The retail company's predicated data can be used to optimise labor utilisation, save inventory costs, and schedule workers during times of high demand. In a few researches, the ARIMA time-series model is also observed to be integrated with the Long Short Term Memory (LSTM) neural network models. These studies' findings demonstrate that the deep learning approach enables businesses to make data-driven decisions more effectively, which boosts workforce planning and management (Sultan et al., 2020; Ramos et al., 2015).

However, when the data is non-linear, there are some challenges that may arise with ARIMA models. In these situations, ”the classical ARIMA approach becomes prohibitive, and in many cases, it is impossible to determine a model, when seasonal adjustment order is high, or its diagnostics fail to indicate that time-series is stationary after seasonal adjustment.” (Fattah et al., 2018).

2.2. Transformer-based Model

In terms of the various neural network types used for time-series prediction, Long Short-Term Memory (LSTM) models are well known and acknowledged as a sophisticated RNN model for predicting nonlinear dynamics by enhancing gradient flow in the networks (Hochreiter & Schmidhuber, 1997; Menculini et al., 2021). It is especially effective at capturing sequential information because it makes use of memory cells that can store data over time (De Saa & Ranathunga, 2020). However, one flaw of Long Short Term Memory (LSTM) is that it is seen to suffer from “short term memory” over long sequence. In theory, LSTM could propagate over infinitely long sequences, but in reality, the LSTM model would forget the earlier data due to the vanishing gradient situation (shown in figure 1). Instead, the transformer model is capable to retain direct connections to all previous timestamps and retain longer sequences through its attention mechanisms (shown in figure 1).

The transformer model, which was first presented in 2017, employs a cutting-edge ”encoder-decoder” architecture with a self-attention mechanism and has gained popularity for a variety of sequence-to-sequence tasks such as language translation (Vaswani et al., 2017; Li et al., 2019). The original transformer model is a traditional sequence-to-sequence model, achieving higher reputation in improving the performance natural language processing, in which the input sequence can have an impact on any portion of the output sequence and the output sequence's length can differ from the input sequence's length. The original transformer model performs better in natural language-created tasks like machine translations based on the encoder-decoder attention mechanism. Time series forecasting, however, is not equivalent to natural language processing. Instead of an-

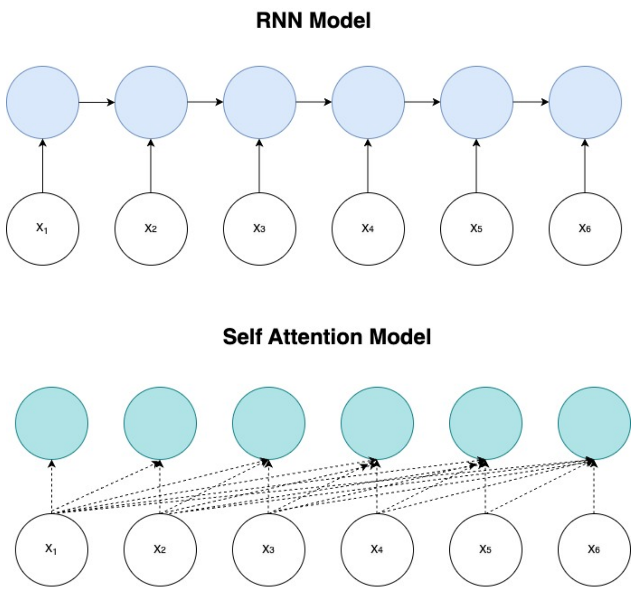

Figure 1. The mechanism behind RNN model and self-attention mechanism, inspired by (Zhang et al., 2021)

ticipating different lengths of a series, it predicts a single step using previous timestamps. Although time series prediction can occasionally be used to forecast multiple steps, it is still conducted in an auto-regressive manner, meaning that the model will first predict one step, then incorporate the previous prediction into the input data and predict the following step, before combining the results. The transformer model still excels in time series prediction because of its self-attention mechanism that propagates longer data sequences and proactively filters out less crucial information to prevent overloading the analysis process.

The self-attention process is described as follows (Klingenbrunn, 2022):

Step 1: Query, key and value vectors creation: To determine their relevance with regard to the present token, the query of the current token is compared to the keys of all other tokens. The new encoding is made using the value vector, which is the actual representation of a given token. Each token is multiplied with three separate weight matrices (for the key, query, and value), which are discovered throughout the model's training process, to perform the necessary transformations.

Step 2: The self-attention score calculation: The self-attention score are computed for each token in the sequence and evaluates how relevant the present token is to any other token that has been observed in the sequence. High ratings suggest a high level of relevance, whilst low scores denote the opposite.

Step 3: Consequently encode the current token: Once the scores from the previous phase are passed into the softmax function, which normalizes the scores into a probability distribution summing to 1. Thus, data with the higher softmax value will be given more weight in the encoding process, while the data with the lower softmax value will contribute less.

Wu et al. (2020) discovered that the Transformer model outperforms other common time-series predictions, such as ARIMA and RNN, and achieves superior accuracy due to its capacity to simulate complicated dynamics of time-series data (Wu et al., 2020). Nowadays, several academics attempt to apply transformer architecture for multivariate time-series data. Transformer designs are used in Zerveas et al. (2021)'s unsupervised technique for multivariate time-series regression and classification (Zerveas et al., 2021). Additionally, Cai et al. (2020) identify the limitations of parallelizing and periodicity forecasting of RNNs, and they suggest the traffic transformer model better captures the continuity of time-series data (Cai et al., 2020).

2.3. Discussion

To sum up, it is clear from the literature that ARIMA models have long been recognized as a popular method for time-series forecasting due to their ease of use and accuracy. The ARIMA model is typically the first step in a project, with further deep learning techniques added as necessary to account for non-linear data. In contrast, transformer-based models have become more popular in recent years, especially in language models but also increasingly for time-series, as they are able to handle non-stationary time data in a way that ARIMA cannot. Consequently, problems with seasonal patterns — which are very prevalent in the retail environment — are more likely to be attracted to this architecture. The outcomes clearly demonstrated that the contemporary version outperformed ARIMA. Transformer-based models can be a powerful tool for any industry because of the vast application potential they possess and their well-performing attention mechanisms. However, it is crucial to build the proper transformer architecture based on the data's properties and the application's requirements.

3. The Architecture

3.1. ARIMA - Baseline Model

An implementation of ARIMA is constructed as means of a baseline for a relative comparison to the transformer-based models. As a statistical model, its complexity is derived from its properties p, d, and q. p is the number of lag observations included, also called lag order, d is the degree of differencing, which is the number of times the observations are differenced, and q is the size of the moving average window. The forecasting equation is constructed as follows. First, let *y* denote the *dth* difference of *Y*which means:

If $d = 0 : yt = Yt$

If $d = 1 : yt = Yt - Yt - 1$

If $d = 2 : yt = (Yt - Yt - 1) - (Yt - 1 - Yt - 2) = Yt - 2Yt - 1 + Yt - 2$

Determining the best values for the model was done through looping over multiple possible values in the range of 1 - 5. Through this experimental procedure the model sought to best capture the trends of the test data with values of *p* = 10, *d* = 3 , and *q* = 3. Further explanation of the results are done in the section below.

3.2. Transformer Architectures

For the transformer-based models, we constructed and experimented with multiple architectures to see how different configurations were able to understand the complexity in the time series data. All models built around the *Keras MultiHeadAttention* layer. First, we developed a simple architecture with a single MultiHeadAttention layer to see the prediction quality at minimal complexity. Second, we constructed an architecture with more prominent encoder and decoder blocks, resembling the original transformer architecture to leverage the encoder-decoder attention.

3.3. Model1 - Simple Transformer Architecture

The simple architecture consists of one input layer, *one* MultiHeadAttention layer followed by one *Dropout* layer and *Normaization* layer, followed by two fully connected *Dense layers* which introduces non-linearity to the model. Finally, it consists of an Output layer which is a Dense layer with a single unit.

The architecture of the basic transformer model is produced as follows:

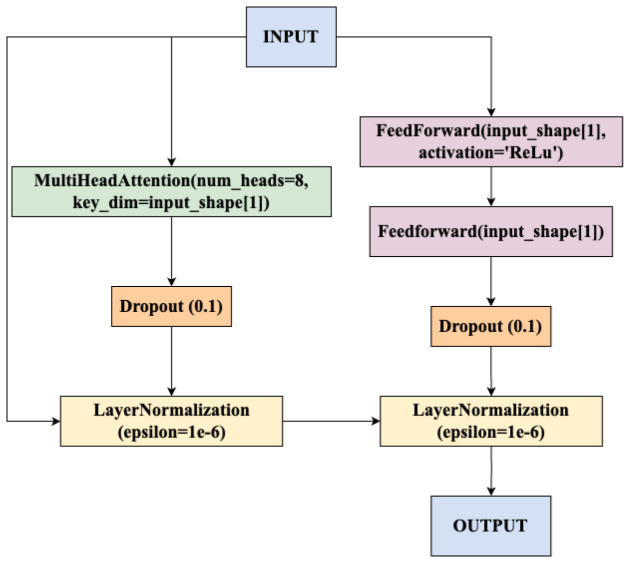
INPUT
MultiHeadAttention(num_heads=8, key_dim=input_shape[1])
Dropout (0.1)
LayerNormalization (epsilon=1e-6)
FeedForward(input_shape[1], activation='ReLu')
Feedforward(input_shape[1])
Dropout (0.1)
LayerNormalization (epsilon=1e-6)
OUTPUT

Figure 2. Simple Transformer Model Architecture

First, the self-attention mechanism represented by the MultiHeadAttention layer aids the model in learning relationships between various input components. The crucial parameter for this layer is *num_heads* which is the number of attention heads for the layer. This is commonly set to either 8 or 16. We did not experience major performance differences between the two and kept it at 8. The following dropout layers implement regularization to prevent overfitting and we set the value to 10%. The normalization layer normalizes the inputs across the features.

Second, the output of the first block gets fed into the feedforward neural network (FFN). The FFN, being expressed by the two Dense layers, applies non-linear transformations to the self-attention output to produce the final prediction output.

This architecture is a simplified version of the conventional transformer, as the first block can be thought of as an encoder. Additionally, the FFN do transform the output of the self-attention mechanism but lacks the typical characteristics of the decoder.

3.4. Model2 - Complex Transformer Architecture

The sophisticated version of the transformer architecture, which draws inspiration from the traditional transformer architecture, adds additional layers and creates more distinguished encoder-decoder attention mechanism in hopes for better capturing the inherent complexities in the time-series data.

The design of the complex transformer model is as follows:

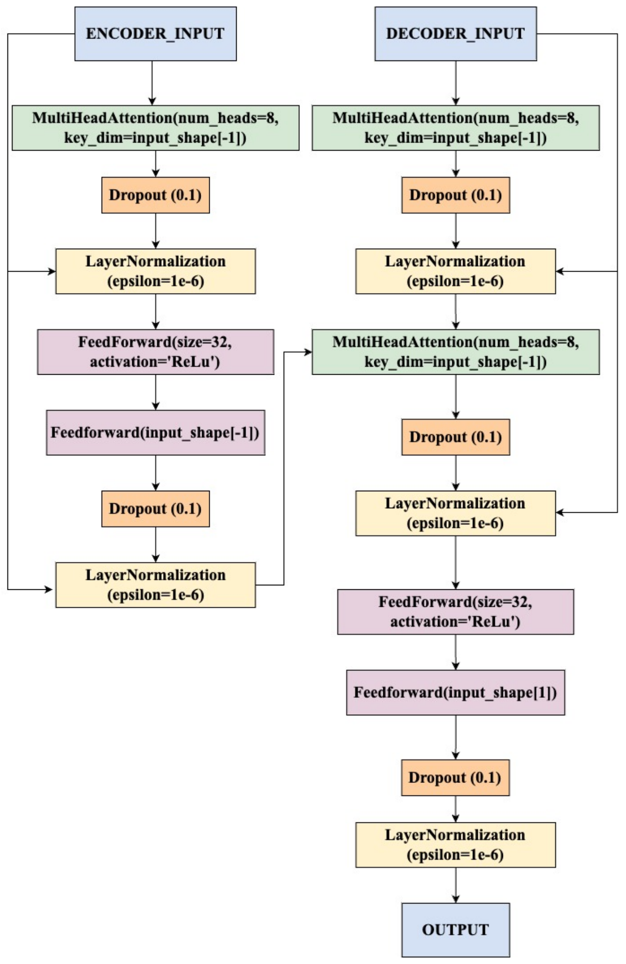
ENCODER_INPUT
DECODER_INPUT
MultiHeadAttention(num_heads=8, key_dim=input_shape[-1])
Dropout (0.1)
LayerNormalization (epsilon=1e-6)
FeedForward(size=32, activation='ReLu')
Feedforward(input_shape[-1])
Dropout (0.1)
LayerNormalization (epsilon=1e-6)
MultiHeadAttention(num_heads=8, key_dim=input_shape[-1])
Dropout (0.1)
LayerNormalization (epsilon=1e-6)
FeedForward(size=32, activation='ReLu')
Feedforward(input_shape[1])
Dropout (0.1)
LayerNormalization (epsilon=1e-6)
OUTPUT

Figure 3. Complex Transformer Model Architecture

The structure of the complex transformer model consists of three main sections: *encoder self-attention*, *decoder self-attention*, and *encoder-decoder attention*. For expressing self-attention, the encoder block uses one MultiHeadAttention layer, which correlates distinct places in the input sequence and highlights the sections that are most important to one another for further processing. Applying a feed-forward neural network after a self-attention mechanism allows the output to shift from linear to non-linear relationships so as to capture more intricate non-linear relationships among the input sequence's component elements. Like the encoder, the decoder's self-attention component focuses on various input sequence parts to produce outputs. Then, it the output from the decoder self-attention is combined with the encoder output to generate the final output of the decoder block. This creates the valuable encoder-decoder attention which ties the encoder and decoder together. This is then passed through Dropout-regularization and normalization before being fed to the decoder FFN.

4. Training Methods

This section covers the training methods for the architectures used in the experiments and the oprimizers and hyperparameters. For the ARIMA model, there is not much relevance to discuss the training methods, however, it got trained on 80% of the data and the parameters got set to the values explained above.

For the transformer based models, the training methods got executed in very similar ways. The main difference being that Model2 and with its two inputs to both the encoder and the decoder block, it requires two input values for the input data parameters when calling the *model.fit()* method. The same is true for the parameter *validation_data*. On the other hand, the simpler Model1 requires only one value as input data and validation data. The second difference is that Model1 is trained over 25 epochs compared to 50 epochs for Model2. The reason being the complex structure requires more training in order to grasp the concepts in the data. It should be noted that we ran Model1 over 50 epochs as well without seeing any increase in accuracy compared to 25. Both models use 80% of the data for training and 20% for test.

Another important notion is that both transformer-based models are trained with the variable $windowsize = 1$ meaning that the only the current time step is used to predict the value of the next. As a third experiment, we explored with increasing this parameter, however, without gaining higher performance. This is further explained in *Section 5.3*.

4.1. Optimizer

Both Model1 and Model2 uses the Adam optimizer for training. Adam is a stochastic gradient descent (SGD) algorithm based on adaptive estimation of first-order and second-order moments. Adam combines characteristics of AdaGrad and RMSProp algorithms to better handle sparse gradients. According to the original paper proposing the algorithm, Adam is "computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data/parameters" (Kingma & Ba, 2014). The learning rate of: $\epsilon = 0.001$. The loss is set to Mean Squared Error (MSE) given the nature of the regressional problem of the time-series prediction.

4.2. Other Training Parameters

The *batch size* for both Model1 and Model1 is set to 1, meaning that the model is updated after each training instance, known as stochastic training. This has the possibility to increase the required training time, however, it can increase the models ability to generalize. During exploration, we explored with values such as 32 and 64 without seeing any performance increase.

4.3. Measures to Prevent Overfitting

As covered when going through the model architectures, a measure to prevent the model from overfitting is the implementation of multiple Dropout layers. As the data is not overly complex, this assures that the sophisticated nature of the transformer models do not overfit the data. Setting the dropout value to 10 makes sure that 10% of output values of the layer gets ignored at random.

5. Numerical Results

5.1. Goals & Data Preparation

The goal of the time-series prediction is to find the best performing model to predict the future staffing for the business. This includes a low as possible prediction error on the test data. More precisely, the performance capabilities of the ARIMA model will be used as a baseline to compare the simple transformer model (Model1), and the complicated transformer model (Model2). The key comparison metric for assessing the performance of various models would be *model loss* (square root of the MSE (RMSE) to achieve comparable scale to original data). Achieving lower loss on the test set indicates higher predictive capabilities.

The raw dataset is derived from a real-world retailing business and consists of features explaining the number of employees working and number of consumers visiting in 30-minute time-slots. This generates roughly 1500 rows when aggregated to whole days (just under 4 years of data). *Table 1* shows sample data from the dataset and the respective features.

Table 1. Snippet of the dataset.

TIME_SLOT	N_VISIT	N_SHIFTS	UTILISATION
2019-01-14	14	36	0.600023
2019-01-15	24	36	0.651213
2019-01-16	24	36	0.665499
2019-01-30	23	36	0.661332
2019-01-31	22	37	0.680456

N_visits is the feature covering the number of customer visits that day, *N_shifts* shows the number of 30-minute employee time slots a day at that one retail location, and *Utilization* is the two former divided to get a measure of the staffing efficiency. *Timeslot* is the index of the dataframe. Utilization is the feature we want to predict, using n_visits and n_shifts as input features.

Figure 4 visualises the trend of the Utilization feature over time. As a part of data preparation, some small time periods at the beginning and the end of the dataset got deleted, as they contained above-normal mounts of noise.

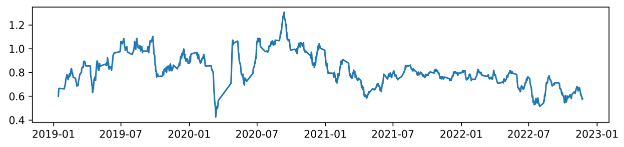

Figure 4. Data visualization after cleaning

5.2. Baseline Model-ARIMA

The ARIMA time-series prediction model is chosen as the baseline method. To evaluate the effectiveness of the ARIMA model, the different value between 1 and 5 that represents the smoothness of the model is selected. As clearly displayed in *Figure 5* the model is vary able to fit the training data, however, it makes very bad predictions on the test data. This could indicate a sign of the model overfitting the training data even though it has a simple structure. The best performing ARIMA model scored a $RMSE = 0.717$ which is high given the numeric scale of the data.

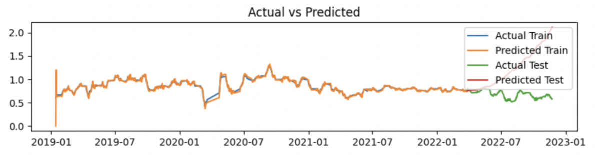

Figure 5. Result of ARIMA Model

5.3. Transformer Model Comparison

In order to find a better forecasting performance for the transformer model, we compare the simpler Model1 as well as the more complex Model2. Both of these are trained with the variable $windowsize = 1$. The comparisons will be completed in two steps. First, the results of Model1 and Model2 the fixed window size variable fixed as 1 are compared. Second, these results are compared to the results from increasing the window size of the transformer.

5.3.1. Model1 - Simple Transformer Model (window size=1)

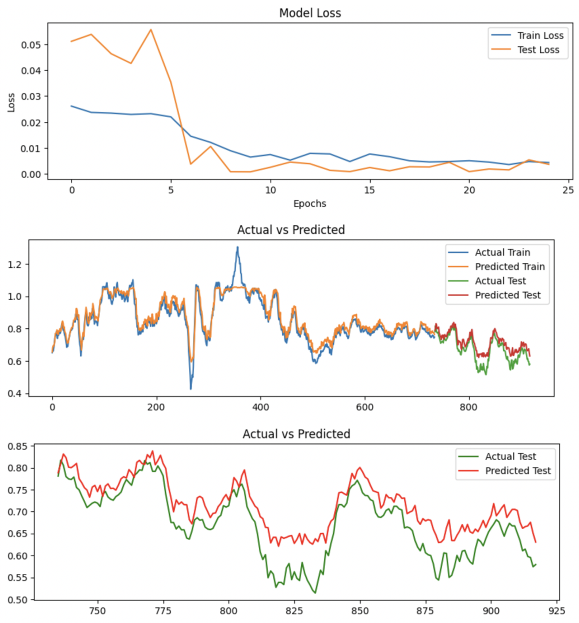

Figure 6. Result of Simple Transformer Model

The simple transformer architecture is performing fairly good results on the time-series data, both on the train and test data. The model converges quickly and does not seem to overfit the data, as the validation loss does not exceed the training loss in *Figure 6*. This is also shows as the model does not fit the largest outlier peaks of the training data. The model achieved the values: $TrainingRMSE = 0.04$ and $TestRMSE = 0.05$ on predictions.

5.3.2. Model2 - Complex Transformer Model (window size=1)

According to the findings, the Model2 outperforms the simple model on both training and test predictions. As seen in *Figure 7* the model does not overfit either, and converges at 50 epochs. The models fit to the training data is slightly better with a validation loss of 0.0018 during training. Model2 achieved the values: $TrainingRMSE = 0.04$ and $TestRMSE = 0.03$ on predictions, slightly lower than the simple model.

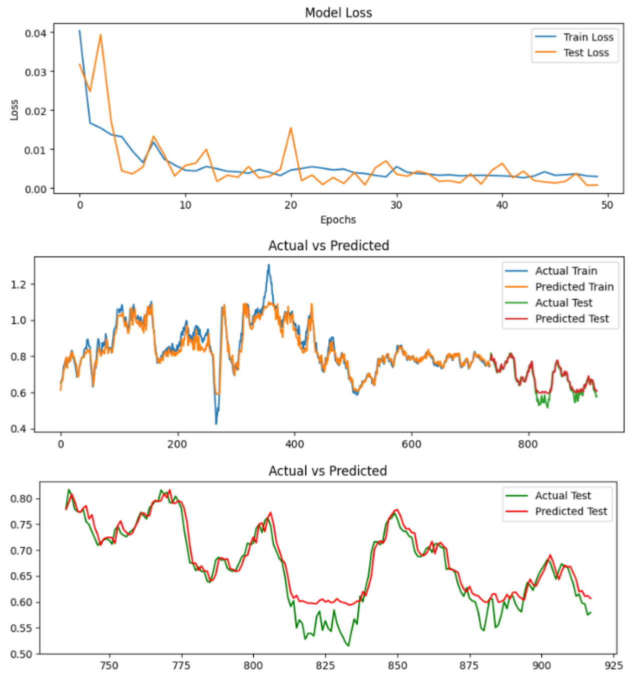

Figure 7. Result of Complex Transformer Model (window size=1)

5.3.3. Model2 - Complex Transformer Model (*look_back*=30)

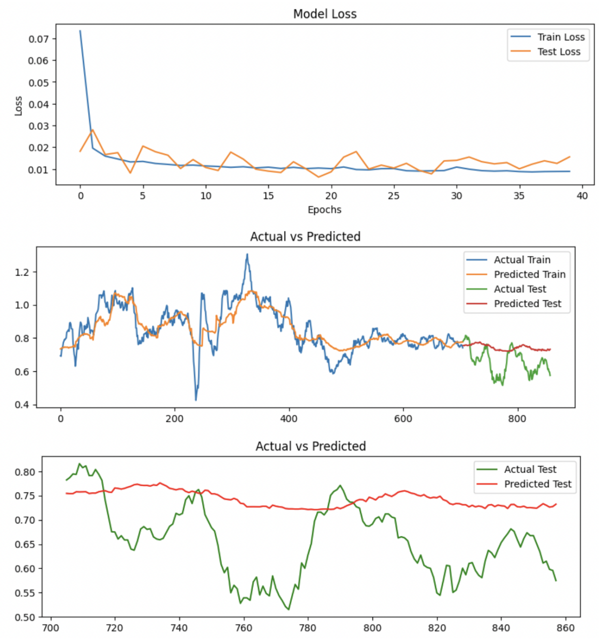

Figure 8. Result of Complex Transformer Model with *look_back*=30

As a last experiment, we increased the window size parameter to 30 as this enables the model to analyze larger time frames at once and better grasp the inherent complexities in the data. However, this change does not better the performance at all, and makes it way less capable to accurately predict the time-series, as seen in *Figure 8*. However, the model very quickly converged, arguably before 10 epochs. As numerical results, the model achieved the values: $TrainingRMSE = 0.08$ and $TestRMSE = 0.11$ on predictions, making it the lowest performer out of the transformer based models.

5.4. Performance Discussion

It is interesting to notice that the ARIMA model, even though being recognized as a well performing time-series prediction model in the literature, is highly inefficient to predict future tendencies on our dataset. With the best performing parameter configurations it was able to predict the general trends of the test set, but without any details. Thus, all transformer-based models outperformed the baseline, which is a promising sign for deep learning algorithms, however, it was to be expected given the vast difference in sophistication.

When it comes to the transoformer-based models it is interesting to note that the simple architecture of Model1 is very capable of comprehending the patterns in the data. Even with only one MultiHeadAttention layer we argue that the results are of acceptable of performance, and given its need for lower training time and the general lowed complexity, it should be considered a good alternative for time-series prediction of this nature.

The most complex architecture with clearly defined encoder and decoder blocks performs the best out of them all. This proves that the original transformer architecture provided in 2017 is highly capable of tasks outside of language processing, and should be seen as a very sufficient neural network architecture for time-series prediction. Last, it is interesting to see that with the increased window size, the model takes in larger amounts of sequences at once during training, but is not capable of making any better predictions. However, one should not exclude such adjustments for time-series problems as it may differ depending on the problem at hand.

6. Conclusion

To conclude, this study aimed to compare the performance of different time-series prediction, namely the ARIMA model, with the modern transformer model. Both models according to the literature, and to our own results have proven to be effective for workforce optimization in retail environments but each to a different degree. While the ARIMA model is the more classical and traditional statistical model, it can only be used as a baseline due to its lack of complexity and flexibility. The models created test on model loss, where a lower loss indicates better predictive performance. Using our dataset formulated for a real-world retailing company, we concluded that ARIMA determines the best values for the model only by looping over multiple possible values. On the other hand, the modern transformer model, inspired by the traditional transformer architecture, adds additional layers, and creates a mechanism that can test long-term dependencies and relationships between the input components where ARIMA struggles with those. The sophisticated transformer model, is more powerful and achieved the lowest model loss and thus, proved efficient in forecasting workforce optimization.

While the study efficiently compares different time series predictions and decides on which one performs best, there are several limitations that should be considered. Firstly, the study bases the results on one specific retail company and the findings may not be able to be generalized to other industries or other retail companies since there can be a lot of other external factors in play. Secondly, the study only compares the performance of ARIMA and transformers with a fixed set of hyperparameters on one dataset. And lastly, the results are concentrated on a specific performance metric, not considering any others such as computation time or interpretability which may be useful in a real-world application.

Consequently, some possible areas for future research include testing the performance of the transformer on different types of datasets or compare them to other modern models to widen the results pool. It would also be interesting to investigate the impact of different hyperparameters on the performance of the transformer model while also tweaking the performance metrics for further analysis.

References

Amar, J., Rahimi, S., Bismarck, N. v., and Wunnava, A. Smart scheduling: How to solve workforce-planning challenges with AI. https://www.mckinsey.com/capabilities/operations/our-insights/smart-scheduling-how-to-solve-workforce-planning-challenges-with-ai, nov 1 2022. [Online; accessed 2023-04-29].

Cai, L., Janowicz, K., Mai, G., Yan, B., and Zhu, R. Traffic transformer: Capturing the continuity and periodicity of time series for traffic forecasting. *Transactions in GIS*, 24(3):736–755, 6 2020.

De Saa, E. and Ranathunga, L. Comparison between arima and deep learning models for temperature forecasting. *arXiv preprint arXiv:2011.04452*, 2020.

Fattah, J., Ezzine, L., Aman, Z., El Moussami, H., and Lachhab, A. Forecasting of demand using arima model. *International Journal of Engineering Business Management*, 10:1847979018808673, 2018.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Klingenbrunn, N. Transformer Implementation for Time-series Forecasting. https://medium.com/mlearning-ai/transformer-implementation-for-time-series-forecasting-a9db2db5c820, apr 10 2022.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

Mao, L., Huang, Y., Zhang, X., Li, S., and Huang, X. Arima model forecasting analysis of the prices of multiple vegetables under the impact of the COVID-19. *PLOS ONE*, 17(7):e0271594, jul 28 2022.

Menculini, L., Marini, A., Proietti, M., Garinei, A., Bozza, A., Moretti, C., and Marconi, M. Comparing Prophet and Deep Learning to ARIMA in Forecasting Wholesale Food Prices. *Forecasting*, 3(3):644–662, sep 15 2021.

Montero-Manso, P. and Hyndman, R. J. Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*, 37 (4):1632–1653, 10 2021.

Ramos, P., Santos, N., and Rebelo, R. Performance of state space and ARIMA models for consumer retail sales forecasting. *Robotics and Computer-Integrated Manufacturing*, 34:151–163, 8 2015.

Sultan, M. F., Jabeen, M., and Mannan, M. A. Sentiment Analysis through Big Data in online Retail Industry: A Conceptual Quantitative Study on linkage of Big-Data and Assortment Proactive of Online Retailers. *KIET Journal of Computing and Information Sciences*, 3(2):16, jul 2 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wu, N., Green, B., Ben, X., and O'Banion, S. Deep transformer models for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317*, 2020.

Zerveas, G., Jayaraman, S., Patel, D., Bhamidipaty, A., and Eickhoff, C. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2114–2124, 2021.

Zhang, A., Lipton, Z. C., Li, M., and Smola, A. J. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

A. You *can* have an appendix here.

Statement about individual contributions

56071 – Introduction,Related Work, Conclusion Writing

41538 – Architecture, Numerical result, Introduction, Related Work writing graph diagram drawing, Complex Transformer Architecture building, paper polishing

50507 – Data Preparation, ARIMA model building, simple complex transformer architecture building, training method writing, paper polishing.