Inventory Time Series Forecasting with Deep Learning

Project ID: 59

Authors: Manisha Pillai, Daniel Yang, Billy Ge, Zhiming Fan, Zikai Liu

Advisor: Yunkai Zhang

TABLE OF CONTENTS

I. EXECUTIVE SUMMARY

This project explores inventory demand forecasting in the retail sector through the lens of deep learning, using the M5 Kaggle competition dataset from Walmart. Forecasting future demand plays a pivotal role in optimizing inventory levels, mitigating losses from overstocking and understocking, and enhancing supply chain responsiveness. Our team implemented the Encoder-Decoder Transformer and DeepAR models and compared their performances against linear and Encoder-only transformers. By conducting extensive hyperparameter tuning and evaluation using mean absolute error (MAE), we were able to assess trade-offs in accuracy, architectural complexity, and computational efficiency.

Our results indicate that deeper Encoder-Decoder Transformers outperform simpler architectures, while the DeepAR model with teacher forcing achieved competitive results with better efficiency than ancestral sampling. With the help of the real-time monitoring platform Weights & Biases, we learned the critical impact of architectural depth, learning rate, and model regularization strategies on time series forecast accuracy. Additionally, we gained insight into the limitations of deep learning models when applied to highly sparse, hierarchically structured retail data.

II. INTRODUCTION

Accurate inventory forecasting is a critical challenge in large-scale retail, where even a slight prediction error can lead to millions of dollars in losses due to overstocking or understocking. Among various time series forecasting applications, inventory prediction has consistently gained the most attention, as it directly impacts profitability, operational efficiency, and customer satisfaction. Overstocking results in wasted resources and increased storage costs, while understocking leads to lost sales and disappointed customers. For large retailers like Walmart, which operate on a massive scale with thousands of products, even a tiny forecasting error can translate into millions of dollars in losses [1]. By accurately predicting future demand, businesses can optimize inventory levels, reduce waste, and enhance supply chain efficiency. In highly competitive markets, effective inventory forecasting provides a strategic advantage, ensuring that businesses meet customer needs while minimizing excess costs.

A time series is a sequence of data points collected over regular time intervals. Unlike the everyday tasks we solve using machine learning, such as prediction and classification, in which we are interested in the associations between various independent features and the dependent variable, time series forecasting aims to forecast future outcomes based on historical data that follows a sequential pattern over time. Regardless of the methods to solve the time series forecasting problem, the key is to capture temporal dependencies, trends, and seasonality in past data. This unique nature of time series forecasting makes it essential for use cases where time-based patterns and fluctuations play a critical role in decision-making.

Time series forecasting has various applications. In retail, it is applied to inventory forecasting, which helps retailers stock the right products at the right time. In meteorology, it assists in predicting weather conditions, leading to better disaster preparedness. Utilities use time series forecasting in the energy sector to anticipate electricity demand and provide a stable power supply. In finance, traders model time series to predict stock prices. With such diverse uses, traditional models often struggle to capture the complexity and irregularity of real-world data, especially in retail environments affected by seasonality.

Retail forecasting presents several challenges. First, seasonality can lead to fluctuations in inventory demand based on recurring patterns such as holidays, Black Friday, Super Bowl, etc. Second, external factors such as promotions, competitor actions, etc., may also lead to sudden and unpredictable shifts in demand. Fortunately, a rich set of methods has been proposed to perform accurate time series forecasting, ranging from classical autoregressive models to more advanced deep learning models, including probabilistic forecasting, transformer-based models, hybrid architectures, and innovative large language model (LLM) approaches.

To address these challenges, we explore deep learning approaches, specifically the Encoder-Decoder Transformer and DeepAR models. These models are designed to capture non-linear patterns and provide more flexible forecasting capabilities than classical techniques. We implemented both models from scratch and evaluated their performance on the Walmart M5 unit sales dataset. Our findings show that while both models outperformed the baselines, the Encoder-Decoder Transformer model provided the best results in both accuracy and training efficiency.

III. LITERATURE REVIEW

One of the first successes in deep learning-based approaches in this area is the DeepAR model introduced by Salinas et al. [2]. DeepAR utilizes autoregressive recurrent networks (RNNs) to generate probabilistic forecasts tailored to address supply chain inventory requirements. In particular, the LSTM model's ability to capture both short-term and long-term dependencies makes it especially attractive for inventory forecasting since short-term fluctuations and long-term seasonality or promotional events serve as important signals. Jungbluth and Lederer [3] introduce the DeepCAR method, an extension of DeepAR that explicitly accounts for change points in time series data. Change points (e.g., market disruptions or policy changes) can cause sudden demand variations; the DeepCAR method enhances forecasting accuracy by dynamically adapting to these shifts.

The advent of the Transformer architecture, introduced by Vaswani et al. [4], initiated a paradigm shift in sequence modeling, as evidenced by the success of large language models (LLMs) such as GPT. Unlike traditional RNNs, Transformers rely on self-attention mechanisms to capture long-range dependencies. Subsequent adaptations have aimed to apply this fundamental technique to time series forecasting. Das et al. [5] propose a decoder-only foundation model for time series forecasting that borrows concepts from natural language processing. Their model demonstrates competitive zero-shot performance across diverse datasets by segmenting time series data into patches analogous to tokens. This approach challenges traditional specialized

models for domain-specific forecasting, suggesting instead that general-purpose architectures can be fine-tuned for the task.

Wang et al. [6] introduce a Transformer-inspired lightweight model, which replaces the conventional self-attention mechanism with a CNN-based alternative that leverages depthwise convolutions and multi-scale feature extraction. Driven by the success of the Bidirectional Encoder Representations from Transformers (BERT)—notably its masked self-supervised training objective and ability to generate rich contextual embeddings—Wang et al. [7] propose a novel methodology that employs masked autoencoding with Transformer architectures for multivariate time series imputation. This method aims to uncover the multivariate correlations inherent in time series data. Although the dataset (M5) we used is relatively complete, the imputation framework outlined in [7] provides valuable insights for extending forecasting models to more challenging, imperfect datasets. In addition, the proposed approach further enhances model performance by learning with joint loss objectives at the store and product levels of time series data. However, some studies have questioned the effectiveness of Transformer-based methods in time series forecasting. Zeng et al. [8] offer a counter-narrative by demonstrating that, under certain conditions, a simple one-layer linear network can outperform complex Transformer models across multiple datasets.

An emerging area of research is the application of LLMs, which obviate the need for extensive parameter tuning inherent in traditional parametric models. For example, Tang et al. [9] explore this novel paradigm by adapting LLMs with appropriate prompting strategies for forecasting. By tokenizing and embedding time series data supplemented with natural language instructions, they

empirically demonstrate that state-of-the-art LLMs, such as Llama-2 and GPT-4, can yield performance comparable to that of deep learning–based approaches on specific datasets.

Song et al. [10] comprehensively review deep learning-based time series forecasting. The review synthesizes a wide array of methodologies, including DeepAR and Transformer-based approaches, evaluating their strengths and limitations within the context of product demand forecasting.

In addition to laboratory advancements, the M5 Forecasting competition, hosted on Kaggle in 2020 in collaboration with Walmart, has become a landmark benchmark for large-scale retail demand forecasting. The competition focuses on predicting the next 28 days of unit sales for over 30,000 products across 10 Walmart stores in three U.S. states (CA, TX, and WI), using historical daily sales, price, and calendar data. Its evaluation metric, Weighted Root Mean Squared Scaled Error (WRMSSE), accounts for forecasting accuracy across hierarchical levels such as item, department, category, store, and state, thereby emphasizing both local and aggregated performance.

Due to the hierarchical and sparse nature of the M5 dataset, top-performing solutions generally incorporate extensive feature engineering, including calendar-based features (e.g., day-of-week, event flags), price lags, and rolling statistics (e.g., moving averages, momentum). Many competitors employ gradient boosting decision tree (GBDT) models, particularly LightGBM, due to their robustness and scalability in handling tabular data with mixed frequencies and

missing values. To capture temporal patterns, several teams introduce lag features and multi-scale aggregation strategies to bridge short- and long-term dependencies.

Notably, although several participants experimented with deep learning methods—including RNNs, CNNs, and Transformer-based architectures—these were often challenged by the long training times, the need for extensive tuning, and the tabular structure of the input data. As a result, simpler tree-based models often achieved superior performance when coupled with careful feature engineering. Moreover, ensemble strategies, such as model averaging, blending LightGBM with elastic nets or deep nets, and multi-stage pipelines (e.g., classification followed by regression for zero vs. non-zero demand), proved to be especially effective. Notably, many top solutions aligned their training-validation split with the WRMSSE evaluation by adopting custom validation windows and per-level loss weighting. In summary, the M5 competition serves as a valuable empirical benchmark that complements academic developments such as DeepAR, Transformer, and LLM-based approaches for inventory forecasting purposes.

IV. METHODOLOGY

We used the M5 dataset for forecasting to estimate unit sales of Walmart retail goods across various locations over two 28-day forecasting horizons. The dataset comprises three primary files: sales_train.csv, which contains historical daily unit sales data for 3,049 products across 10 stores; calendar.csv, providing temporal context including dates, weekdays, months, and significant events (e.g., Super Bowl, Valentine's Day, and Orthodox Easter); and sell_prices.csv, detailing the weekly sale prices of these products. Importantly, the M5 dataset is hierarchically

structured, allowing aggregation at multiple levels (e.g., product-store, product category, department, and geographical regions), thereby providing a robust platform for evaluating forecasting accuracy across different granularity levels.

Our preprocessing pipeline was designed to handle the large-scale and high-dimensional nature of the dataset while preserving essential information. A key early step involved optimizing memory usage through numeric downcasting. We examined each numeric column's range and downcasted integers and floats to lower-bit representations (e.g., from int64 to int8) using numpy utilities. Categorical variables, such as state code and product categories, were converted from string objects to the pandas category data type, storing only unique category values, hence contributing further to memory savings and subsequent one-hot encoding. During data integration, we computed a new variable, *time_from_start*, derived from the calendar.csv file. This variable represents the number of days elapsed from the beginning of the dataset and serves as the backbone for time series alignment. We integrated the aforementioned processes into a custom PyTorch data loader pipeline, which scales the input features using methods such as *DeepARScaler* for numerical variables and *OneHotScaler* for categorical variables. To streamline the training process, we initially conducted a pilot run on Google Colab for a limited number of epochs, then deployed the full model on a cloud-based GPU instance equipped with a single NVIDIA A100 SXM4 to enable stable, large-scale learning.

We implemented two deep learning models to test (1) an Encoder-Decoder transformer and (2) a DeepAR model:

1. Encoder-Decoder Transformer:

Implementing our transformer model follows the architecture described in Vaswani et al.'s paper [4]. The model consists of an encoder that processes input embeddings and a decoder that outputs the forecast. The encoder and decoder consist of multi-layer self-attention and feedforward networks, while the decoder has additional multi-head attention. We tested a series of hyperparameters, including the number of layers, batch size, learning rate, and number of epochs. Additional tuning would include modifying the dropout rate for regularization and the feed-forward network size.

2. DeepAR model:

   The implementation of our DeepAR model is based on the architecture described by Salinas et al. in their paper proposing the DeepAR model [2]. The model uses LSTM-based recurrent neural networks(RNNs) to model probabilities from time series data. We used a DeepAR model in the hopes that it would better capture seasonal behaviors than the transformer. We then optimized DeepAR performance by tuning the number of RNN layers, learning rate, epochs, and context length. We also implemented teacher forcing, using ground truth values as input to help the model learn more efficiently.

In addition to our two primary models, we implemented two baseline models to contextualize their performance: a simple feedforward linear model and an encoder-only transformer. The linear model provides a minimalist benchmark, while the encoder-only transformer offers insight into how much forecasting capability stems from the encoder alone. We also tuned these models using the same dataset and tracked metrics (MAE) to ensure fair comparison. The encoder-only model retained the transformer encoder architecture but excluded the decoder, projecting the

flattened encoder output directly to forecast values. Both baselines help us interpret the contribution of architectural components and evaluate trade-offs in complexity versus accuracy.

For all models, we framed the forecasting task using a sliding window approach, where each input sequence consists of 90 days of historical data used to predict the next 10 days. This corresponds to a sequence length of 90 and a prediction length of 10. The sequence length determines the context range that the model uses to learn temporal patterns, while the prediction length defines the horizon over which future values are forecasted. These values were chosen to reflect a realistic short-term inventory planning scenario while ensuring enough historical context for model training.

We used the real-time monitoring tool Weights & Biases (WandB) to track runs and keep a running log of model losses, learning rates, GPU usage, and other evaluation metrics to monitor our models' performance. We rented GPU instances from Vast.ai to accelerate training and reduce the time required for experimentation, allowing us to test multiple hyperparameters efficiently. This approach allows us to evaluate the effectiveness of transformer-based and DeepAR models for time series forecasting on the M5 dataset.

Weights & Biases provided a centralized dashboard for tracking all experiments. Key features utilized include:

1. Metric Logging: MAE and MSE were logged alongside training and test losses to compare model performance across runs.

2. System Monitoring: GPU utilization and power consumption were automatically tracked to optimize resource efficiency during training.
3. Visualization Tools: Interactive plots were generated to visualize loss curves, MAE trends (as shown in Figures 1 and 2), and other metrics over training steps. Custom line plots enabled comparisons of different hyperparameter configurations.
4. Run Comparisons: WandB's "Run Comparer" feature was used to analyze the impact of varying encoder-decoder layer counts on forecasting accuracy.

These tools facilitated efficient experimentation by providing real-time insights into model behavior, enabling us to identify optimal configurations quickly while minimizing computational overhead. The visualizations generated by WandB proved invaluable for understanding convergence patterns across different architectures, as seen in Figures 1 and 2.

V. RESULTS

To evaluate the performance of our forecasting models on the M5 dataset, we conducted a series of experiments focusing on hyperparameter tuning. Each experiment isolates the effect of changing specific architectural or training parameters, such as the number of encoder and decoder layers, learning rate, and batch size. The goal of this experimentation is to identify the best-performing configuration for each model type, including the baseline Linear and Encoder-Only Transformer models, as well as our Encoder-Decoder Transformer and DeepAR models. This is done so we can make meaningful comparisons across their optimal setups. By fine-tuning each model and selecting its best variants based on Mean Absolute Error (MAE), we

aim to present a fair and comprehensive benchmark of different deep learning approaches for time series forecasting.

A. Experiment 1: Linear

The linear baseline model served as one of our two reference benchmarks, in which we concatenated all features together into a vector and fed it into a linear layer for retail demand forecasting. Consistent across different learning rates, the linear model achieves a stable MAE performance of 136.29. While straightforward and computationally efficient, its limited capacity to capture complex temporal dependencies resulted in the highest error among all models tested.

B. Experiment 2: Encoder-Only Transformer

In this experiment, we evaluated the performance of an encoder-only transformer architecture for time series forecasting. Unlike the full encoder-decoder transformer, which uses a separate decoder to generate output sequences, the encoder-only variant relies solely on the self-attention mechanism within the encoder. After processing the input sequence, the output embeddings from the encoder are flattened and passed through a linear projection layer to generate predictions for the forecasting horizon. This direct forecasting approach simplifies the architecture and reduces computational complexity, making it a valuable baseline to assess the contribution of the decoder component in forecasting performance.

To understand the behavior of the encoder-only model under different training configurations, we conducted a series of experiments where we varied the number of encoder layers and the learning rate. The number of encoder layers was treated as the primary architectural hyperparameter, reflecting the model's depth and its ability to capture complex temporal patterns. The learning rate was chosen to examine its influence on convergence stability and generalization.

The results, summarized in Table 1, reveal that model performance is highly sensitive to the number of encoder layers used. To isolate the effect of depth, we compared configurations with one, three, and five encoder layers, all using the same learning rate of 0.0005 and a batch size of 256. Among these, the configuration with three encoder layers achieved the best performance, with a mean absolute error (MAE) of 135.60. This slightly outperformed the one-layer variant, which yielded an MAE of 137.12, and substantially outperformed the five-layer model, which produced the highest MAE of 139.84. These findings suggest that while increasing the number of layers from one to three enhances the model's ability to capture more complex temporal dependencies, further increasing the depth to five layers may lead to diminishing returns or even degrade performance, possibly due to overfitting or optimization instability in the absence of a decoder.

Additionally, we analyzed how learning rates affect performance by examining configurations that used a fixed encoder depth of one layer and a consistent batch size of 256 (Table 2). In this set, we varied the learning rate across 0.0001, 0.0005, and 0.001. The model trained with the smallest learning rate, 0.0001, achieved the lowest MAE of 135.78, while performance worsened

as the learning rate increased: MAE rose to 137.12 at 0.0005, and further to 137.88 at 0.001. This trend confirms that shallow encoder-only models benefit from smaller learning rates, which likely stabilize the training dynamics and prevent overshooting during optimization. In contrast, higher learning rates may cause noisy updates that hinder convergence, particularly in simpler architectures with fewer layers to regulate the representation learning.

Number of Encoder Layers	MAE
1	137.12
3	135.60
5	139.84

Table 1: Effects of adjusting number of Encoder layers on MAE for Encoder-Only Transformer

Learning Rate	MAE
0.0001	135.78
0.0005	137.12
0.001	137.88

Table 2: Effects of adjusting learning rates on MAE for Encoder-Only Transformer

C. Experiment 3: Encoder-Decoder Transformer

In this experiment, we explored the impact of varying the number of encoder and decoder layers on the performance of our Encoder-Decoder Transformer model. The primary goal was to optimize the parameters on this architecture for time series forecasting tasks using the M5 dataset. The Mean Absolute Error (MAE) was used as the evaluation metric to measure forecasting accuracy. Table 3 summarizes the results, while Figure 1 illustrates the convergence trends across different configurations. Here, MAE is reported in absolute units of daily unit sales per bottom-level SKU-store series (i.e., average deviation in items sold), rather than as a percentage error.

The results indicate that increasing the number of encoder and decoder layers generally improves performance, as deeper architectures are better equipped to capture complex temporal dependencies. Among the configurations tested, the model with 3 encoder layers and 3 decoder layers (Dark Green) achieved the lowest final MAE of 114.827 at 20K steps, demonstrating its superior ability to model intricate patterns in time series data. Interestingly, configurations with asymmetric layer counts, such as Blue (3 encoder layers, 1 decoder layer) and Red (1 encoder layer, 3 decoder layers), also performed well, suggesting that increasing complexity in either the encoder or decoder can enhance forecasting capabilities. For instance, the Red model achieved a competitive MAE of 118.925, closely matching the performance of Blue (3 encoder layers, 1 decoder layer), which had an MAE of 119.462.

The visualization in Figure 1 provides further insights into training dynamics. Models with fewer layers, such as Light Green (1 encoder layer, 1 decoder layer), exhibited slower but smoother convergence and higher MAE values throughout training. Interestingly, the best performance was

achieved by a moderately deep architecture, the Dark Green (3 encoder layers, 3 decoder layers), which balanced model complexity and stability, resulting in the lowest final MAE. Although the Dark Green model achieved the lowest MAE at the final 20K step, the performance differences between the deeper models were relatively small. For instance, the Orange model (10 encoder layers, 10 decoder layers) briefly achieved the best MAE at 19K steps. This suggests that model performance is close enough that small variations in training can shift the outcome. However, deeper models like Pink and Orange exhibited more fluctuations during training and signs of overfitting in later steps. This suggests that beyond a certain depth, added layers may increase model complexity without providing any meaningful performance gains or accuracy.

Overall, these findings underscore the importance of hyperparameter tuning in optimizing transformer-based models for time series forecasting tasks. While deeper architectures could perform better, the addition of more layers increases the risk of overfitting and increases model complexity without clear improvements in performance. Additionally, configurations with balanced or asymmetric layer counts can also yield competitive results. Future experiments could explore additional parameters, such as attention heads and feedforward network size, to further refine model performance.

Model Color	Number of Encoder Layers	Number of Decoder Layers	MAE
Light Green	1	1	125.642
Blue	3	1	119.462
Red	1	3	118.925
Dark Green	3	3	114.827
Pink	6	6	119.408
Orange	10	10	116.172

Table 3: Performance comparisons across different Encoder-Decoder Transformer architectures

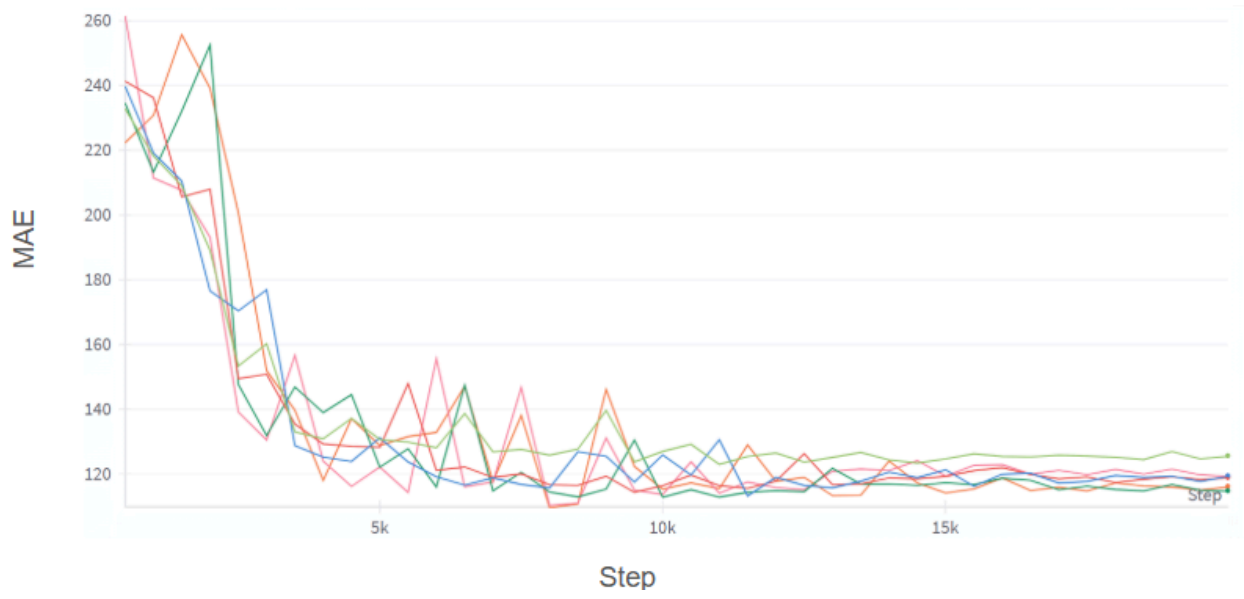

Figure 1: MAE loss convergences of different Encoder-Decoder Transformer architectures

D. Experiment 4: DeepAR

We evaluated multiple variants of the DeepAR implementation on the M5 dataset using mean absolute error (MAE) as the primary performance metric. Two principal approaches were examined: DeepAR with ancestral sampling and DeepAR employing teacher forcing (hereafter "Mean-Teacher Forcing"). These variants were compared under varying encoding hyperparameters, learning rates, and batch sizes.

The ancestral sampling variant, which leverages a Gaussian negative log-likelihood (NLL) loss to simultaneously predict both the mean and standard deviation of the future distribution, achieved MAE values of approximately 213.4 and 209 under different configurations. Despite its capacity for probabilistic forecasting—enabling the generation of multiple sample paths for ensemble averaging—the method's point estimation performance was relatively weaker. Notably, ancestral sampling improved its point forecast MAE by around 12 points when lowering the learning rate (from 0.0005 to 0.0001) and reducing the batch size, suggesting the potential for granular hyperparameter tuning. In contrast, the Mean-Teacher Forcing variant consistently outperformed ancestral sampling on point forecasts. Under the best configuration, DeepAR (Mean-Teacher Forcing) achieved an MAE of 131 when using an encoding hyperparameter of 1 with a learning rate of 0.0001 and a batch size of 32. This observation is in line with prior work suggesting that teacher forcing can be particularly advantageous for iterative models in scenarios where the evaluation criterion targets point accuracy rather than probabilistic calibration.

We also demonstrate that although DeepAR employs a simpler backbone—Long Short-Term Memory (LSTM), a type of Recurrent Neural Network (RNN) designed to mitigate issues like the vanishing gradient problem and effectively capture historical context by processing data sequentially—it can potentially outperform more complex transformer architectures with careful fine-tuning. However, DeepAR's sequential nature inherently restricts parallel computation, resulting in slower training and inference compared to transformer-based models.

Model Color	Teacher Forcing / Ancestral Sampling	Batch Size	Learning Rate	MAE
Purple	TF	256	0.05	171.0657
Blue	AS	256	0.0005	213.4460
Orange	TF	256	0.0005	194.6028
Yellow	TF	512	0.0005	189.1583
Green	TF	128	0.0001	153.4273
Pink	TF	256	0.0001	158.9022
-	TF	32	0.0001	131
-	AS	32	0.0001	209

Table 4: Performance comparisons across different DeepAR architectures

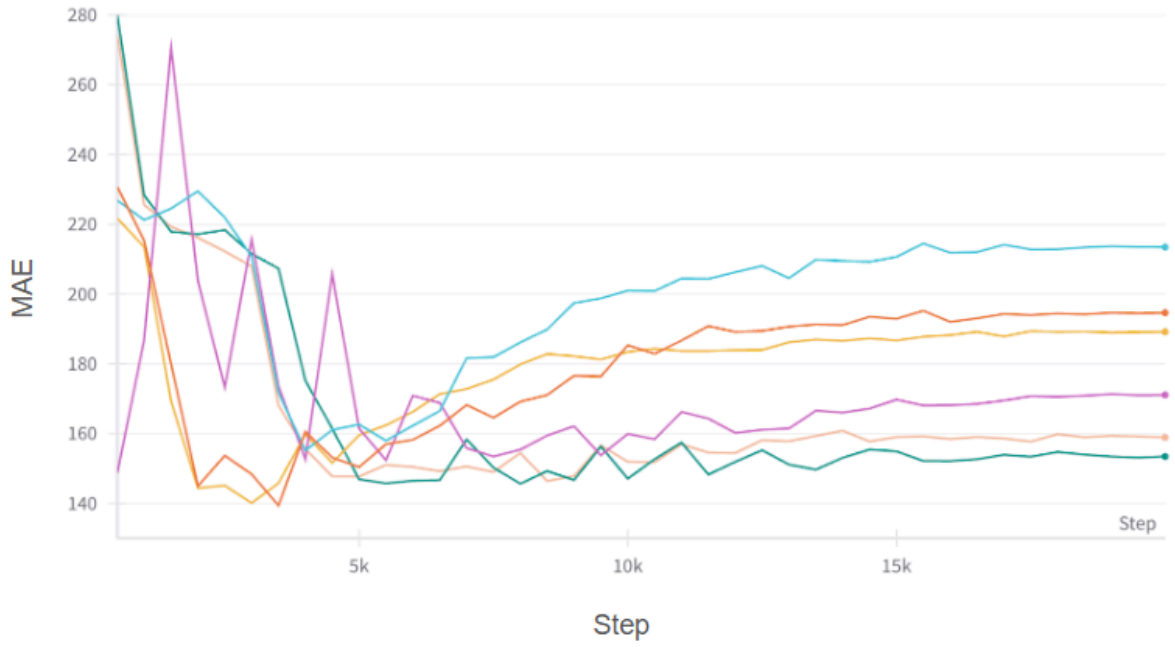

Figure 2: MAE loss convergences of different DeepAR architectures

E. Model Comparison

To provide a comprehensive benchmark of deep learning approaches for time series forecasting on the M5 dataset, we compared the best-performing implementations of the feedforward linear baseline, Encoder-Only Transformer, Encoder-Decoder Transformer, and DeepAR models. We primarily used Mean Absolute Error (MAE) as the comparison metric, with some additional comparisons of model complexity and computational efficiency. The Feedforward Linear baseline served as the base reference point with an optimal MAE of 136.29. While straightforward and computationally efficient, its limited capacity to capture complex temporal dependencies resulted in the highest error among all models tested.

The Encoder-Only Transformer improved upon the linear baseline, with its optimal configuration (three encoder layers, learning rate 0.0005), reaching an MAE of 135.60. This demonstrates that even a reduced transformer architecture relying solely on self-attention within the encoder can effectively model temporal patterns and provide a simpler alternative for direct time series forecasting. Notably, the encoder-only model's simplicity makes it attractive for deployment in more resource-constrained environments, though it does not fully leverage the sequence-to-sequence capabilities of the transformer framework.

The Encoder-Decoder Transformer outperformed both the linear and encoder-only models by a significant margin. Its best configuration—featuring 3 encoder layers and 3 decoder layers—achieved the lowest MAE of 114.827, which outperformed deeper configurations of this model. This highlights the diminishing returns of increasing model complexity, risking overfitting and resulting in lower training efficiency. Furthermore, configurations with asymmetric layer counts (more layers in the decoder than the encoder, for example) also performed competitively, suggesting that increasing complexity in either encoder or decoder components can enhance forecasting accuracy.

The DeepAR model offered a different trade-off. While its ancestral sampling variant lagged with MAEs around 209–213, the Mean-Teacher Forcing variant, when carefully tuned (encoding hyperparameter of 1, learning rate 0.0001, batch size 32), achieved an MAE of 131. By feeding the true target at each time step back into the decoder rather than its own previous prediction, teacher forcing mitigates error accumulation over long horizons—akin to the bullwhip effect in retail, where minor demand fluctuations amplify across the supply chain—and thus yields the

best MAE performance. In contrast, ancestral sampling generates forecast trajectories by sequentially drawing from the learned Gaussian distributions (parameterized by predicted means and variances), but its reliance on Gaussian NLL optimization without conditioning on ground-truth values can lead to higher point-forecast errors unless carefully tuned. Overall, DeepAR generally outperforms the encoder-only transformer but falls short of the encoder-decoder transformer. Its LSTM-based sequential modeling excels at capturing historical dependencies, and with teacher forcing it can approach the accuracy of more complex transformer architectures; however, its inherently sequential nature limits parallelization, leading to slower training and inference compared to transformer-based models.

In summary, the encoder-decoder transformer demonstrated the best forecasting accuracy, followed by DeepAR (with teacher forcing), the encoder-only transformer, and finally, the feedforward linear baseline. These results portray the importance of model architecture and hyperparameter tuning in time series forecasting. Notably, increasing model depth does not always yield better performance, and moderately deep or simpler models strike a better balance between accuracy, generalization, and computational cost.

VI. DISCUSSION

Accurate sales forecasting is pivotal in supply chain management, enabling businesses to optimize inventory levels, reduce costs, and enhance customer satisfaction. In the context of recent egg shortages in the U.S., driven by avian influenza outbreaks and supply chain disruptions, precise demand forecasting could have mitigated the impact by informing proactive

measures such as adjusting procurement strategies and managing inventory more effectively. Implementing advanced forecasting models can thus provide businesses with the agility to respond to market fluctuations and prevent potential losses due to stockouts or overstocking. For a large retailer like Walmart, our findings emphasize the importance of investing in deep learning-based forecasting approaches that can adapt to irregular demand patterns and external events. Our results indicate that the Encoder-Decoder Transformer is the most effective approach, with its 3-layer configuration outperforming deeper variants, delivering higher accuracy at a lower computational cost. By adopting moderately deep Transformer-based forecasting models, Walmart can improve inventory planning, reduce waste, and better align supply with fluctuating customer demand.

VII. CONCLUSION

In this project, we systematically compared several forecasting models and tuning strategies to improve time series prediction for retail demand, focusing on hyperparameter tuning and model architecture optimization. Our experiments revealed that moderately deep encoder-decoder transformer architectures outperformed both shallower and deeper variants, suggesting that increasing model complexity does not always translate to increased model performance. Simpler models like feedforward linear neural networks and encoder-only transformers offered faster training and lower resource demands but were less effective in capturing the intricate temporal dependencies present in retail sales data. The results suggest that careful tuning of both model architecture and training parameters is crucial for maximizing performance. We also show that training techniques like teacher forcing and ancestral sampling can mitigate key retail challenges,

such as the bullwhip effect, and improve safety-stock management. Our work demonstrates the value of advanced machine learning techniques in addressing real-world challenges in retail forecasting.

VIII. FUTURE WORK

To enhance the robustness and applicability of our forecasting models, future efforts will focus on two key areas. First, we aim to fine-tune the DeepAR model with ancestral sampling to better capture the probabilistic distribution of future demand. This approach will enable the estimation of prediction intervals, facilitating the calculation of safety stock levels and informing risk-aware inventory decisions. Second, we plan to explore and compare state-of-the-art transformer architectures such as Informer and DEformer. Informer addresses the challenges of long-sequence forecasting through efficient sparse attention mechanisms, while DEformer employs dual encoders to simultaneously capture temporal and spatial dependencies in multivariate time series data. Additionally, incorporating exogenous variables like store location, weather conditions, and promotional events is expected to improve model accuracy. These advancements hold significant potential for real-world applications in large-scale retail operations, such as those of Amazon and Walmart, by enhancing demand forecasting and inventory management strategies. Future work could also explore how these models can be made more interpretable and adaptable in real-world settings, such as adjusting parameters to account for seasonal trends and sales events.

IX. REFERENCES

[1] G. Mweshi. "Effects of Overstocking and Stockouts on the Manufacturing Sector," in *International Journal of Advances in Engineering and Management*, vol. 4, pp. 1054, 2022.

[2] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, 'DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks', *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[3] Ayla Jungbluth, Johannes Lederer, "The DeepCAR Method: Forecasting Time-Series Data That Have Change Points," 2023. *arXiv,* https://doi.org/10.48550/arXiv.2302.11241. https://doi.org/10.1016/j.ijforecast.2019.07.001.

[4] A. Vaswani, et al., 'Attention is All you Need', *Advances in Neural Information Processing Systems*, 2017, vol. 30. https://doi.org/10.48550/arXiv.1706.03762.

[5] Abhimanyu Das, et al., "A decoder-only foundation model for time-series forecasting," 2024. *arXiv*, https://arxiv.org/abs/2310.10688.

[6] X. Wang, et al., "Transformer-Inspired Lightweight Model for Efficient Time Series Forecasting," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 6655-6659. https://doi.org/10.1109/ICASSP48485.2024.10448163.

[7] Y. Wang, et al., "Multivariate Time Series Imputation Based on Masked Autoencoding with Transformer," in *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application*

(HPCC/DSS/SmartCity/DependSys), 2022, pp. 2110-2117. https://doi.org/10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00313.

[8] Ailing Zeng, et al., "Are Transformers Effective for Time Series Forecasting?," 2022. *arXiv*, https://arxiv.org/abs/2205.13504.

[9] H. Tang, et al., "Time Series Forecasting with LLMs: Understanding and Enhancing Model Capabilities," *SIGKDD Explorations*, vol. 26, no. 2, pp. 109–118, 2025. https://doi.org/10.1145/3715073.3715083.

[10] X. Song, L. Deng, H. Wang, et al., 'Deep Learning-Based Time Series Forecasting,' *Artificial Intelligence Review*, vol. 57, no. 3, pp. 2709–2738, 2024. https://doi.org/10.1007/s10462-024-10989-8.