FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Deep Learning for Demand Forecasting in Food Retail: An Accuracy, Scalability, and Robustness Study

Manuel António Góis Serrano

Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Luís Paulo Reis

Second Supervisor: Prof. Moisés Santos

July 21, 2026

Deep Learning for Demand Forecasting in Food Retail: An Accuracy, Scalability, and Robustness Study

Manuel António Góis Serrano

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. Ricardo Cruz
Examiner: Prof. Paulo Novais
Supervisor: Prof. Luís Paulo Reis

July 21, 2026

Resumo

A previsão de procura no retalho alimentar é um problema tecnicamente exigente, impulsionado pela elevada heterogeneidade dos portefólios de séries loja-produto, pela prevalência de padrões de procura intermitentes e esparsos, e pela volatilidade introduzida por promoções e eventos sazonais. Embora os métodos de aprendizagem profunda tenham demonstrado potencial neste domínio, a evidência empírica sobre o seu valor prático em condições operacionais reais permanece limitada, com poucos estudos a examinar simultaneamente precisão, escalabilidade e robustez de forma integrada. Esta dissertação aborda essa lacuna comparando três arquiteturas globais de aprendizagem profunda treinadas em dados operacionais reais de grande escala provenientes do conjunto de dados M5 da Walmart.

São avaliadas três arquiteturas: uma rede Long Short-Term Memory (LSTM), o Temporal Fusion Transformer (TFT), e uma variante não-autorregressiva do DeepAR designada DeepAR-Direct. Todos os modelos são treinados globalmente sobre 30.490 séries loja-produto, utilizando uma estratégia de previsão direta multi-passo, uma função de perda de desvio de Tweedie, e hiperparâmetros selecionados por optimização Bayesiana com validação cruzada de janela expansiva com quatro partições. A precisão preditiva é avaliada face a modelos de referência estatísticos clássicos e a soluções de topo da competição M5. A escalabilidade é investigada em quatro cenários de portefólio com dimensões entre uma e dez lojas. A robustez é analisada decompondo o erro por série nas quatro classes de padrão de procura do esquema de classificação de Syntetos-Boylan e relacionando a dificuldade de previsão com propriedades das séries temporais extraídas com a biblioteca `ete_ts`.

Os três modelos de aprendizagem profunda superam consistentemente os modelos de referência clássicos e os métodos globais de aprendizagem automática na métrica de precisão ponderada, alcançando melhorias de aproximadamente 23 a 25 por cento face ao modelo de referência Naïve Sazonal e margens de 2 a 3 pontos percentuais sobre os melhores modelos de referência clássicos. Entre as arquiteturas propostas, o TFT alcança a melhor precisão preditiva. A análise de escalabilidade revela que o DeepAR-Direct oferece o equilíbrio mais favorável entre precisão e eficiência computacional, com tempos de treino entre duas e treze vezes inferiores aos do TFT e configurações de hiperparâmetros estáveis ao longo dos diferentes cenários. A análise de robustez identifica a procura Intermitente, que representa 73,3% do portefólio, como a principal fonte de dificuldade de previsão, com as três arquiteturas a degradarem-se de forma semelhante à medida que a procura se torna mais esparsa e irregular. Na classe Intermitente, a flutuação emerge como a propriedade das séries temporais mais fortemente associada ao erro de previsão, com séries de baixa flutuação a contribuírem para a cauda de elevado erro em todos os modelos.

Palavras-chave: Previsão de Procura • Retalho Alimentar • Aprendizagem Profunda • Séries Temporais • Escalabilidade • Robustez

Abstract

Accurate demand forecasting in food retail is a technically demanding problem, driven by the high heterogeneity of store-item portfolios, the prevalence of intermittent and sparse demand patterns, and the volatility introduced by promotions and seasonal events. Although Deep Learning (DL) methods have shown promise in this domain, empirical evidence on their practical value under real operational conditions remains limited, with few studies examining accuracy, scalability, and robustness in a unified framework. This dissertation addresses that gap by comparing three global DL architectures trained on large-scale real operational data from the M5 Walmart dataset.

Three architectures are evaluated: a Long Short-Term Memory (LSTM) network, a Temporal Fusion Transformer (TFT), and a non-autoregressive variant of DeepAR referred to as DeepAR-Direct. All models are trained globally across 30,490 store-item series using a direct multi-step forecasting strategy, a Tweedie deviance loss, and hyperparameters selected through Bayesian optimisation with four-fold expanding window cross-validation. Predictive accuracy is assessed against classical statistical baselines and top-performing M5 competition solutions. Scalability is examined across four portfolio sizes ranging from one to ten stores. Robustness is analysed by decomposing per-series error across the four demand pattern classes of the Syntetos-Boylan framework and relating forecast difficulty to time series properties extracted with the `ete_ts` library.

All three DL models consistently outperform classical statistical baselines and global Machine Learning (ML) methods on the weighted accuracy metric, achieving improvements of approximately 23 to 25 percent over the Seasonal Naïve baseline and margins of 2 to 3 percentage points over the best classical baselines. Among the proposed architectures, the TFT achieves the best predictive accuracy. Scalability analysis reveals that the DeepAR-Direct offers the most favourable balance between accuracy and computational efficiency, with training times between two and thirteen times lower than the TFT and stable hyperparameter configurations across portfolio sizes. The robustness analysis identifies Intermittent demand, which accounts for 73.3% of the portfolio, as the primary source of forecast difficulty, with all three architectures degrading in a broadly parallel fashion as demand becomes sparser and more irregular. Within the Intermittent class, fluctuation emerges as the time series property most strongly associated with forecast error, with low-fluctuation series contributing to the high-error tail across all models.

Keywords: Demand Forecasting • Food Retail • Deep Learning • Time Series • Scalability • Robustness

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework for achieving a better and more sustainable future for all. It includes 17 goals to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

The higher impact of the research presented here is reflected in SDGs where data-driven decision-making and sustainable resource management are directly relevant. By developing and evaluating advanced Deep Learning architectures for demand forecasting in food retail, the findings contribute to reducing food waste along supply chains and to fostering technological innovation in industrial settings, directly supporting the following goals:

SDG 9 promotes resilient infrastructure, inclusive industrialisation, and technological innovation as drivers of sustainable economic development, recognising that scientific research and upgraded technological capabilities are fundamental enablers of industrial progress.

SG 12 promotes sustainable consumption and production patterns, with a central ambition of halving global food waste at the retail and consumer levels by 2030 and decoupling economic growth from environmental degradation.

SDG	Target	Contribution	Performance Indicators and Metrics
9	9.4	The adoption of Deep Learning methods for demand forecasting modernises food retail operations by replacing less efficient forecasting processes, enabling more resource-conscious and sustainable industrial decision-making at scale.	Computational efficiency of developed architectures as measured by training time across forecasting scenarios of increasing scale.
	9.5	Three advanced Deep Learning architectures were developed, evaluated, and benchmarked against classical statistical and Machine Learning baselines on a large-scale real operational dataset.	Forecast accuracy improvement of Deep Learning architectures over classical statistical.
12	12.2	By enabling more accurate demand forecasting at scale, this study supports the reduction of unnecessary food production and associated resource consumption, including energy, water, and raw materials.	Proportion of store-item series for which forecast accuracy surpasses the seasonal naïve baseline.
	12.3	By improving the accuracy of demand forecasting at store-item level, the study directly supports the reduction of food waste at the retail level and along supply chains.	Systematic forecast bias across store-item series as measured by Bias.

Acknowledgements

To Professor Luís Paulo Reis, for giving me the opportunity to carry out this dissertation.

To Professor Moisés Santos, for the close guidance throughout the entire project, for his constant availability and for the rigour with which he supervised me. His suggestions and support were decisive for the development of this dissertation.

To José Borges, for the meetings held over the course of the year and for the contributions he offered, which helped to enrich this work.

To my friends and colleagues who, directly or indirectly, accompanied and supported me throughout this academic journey.

Finally, to my parents, for allowing me to get this far, for always providing me with the best conditions and for their unconditional support throughout my life. To my sister, for her constant presence and for being an example of perseverance and dedication that I strive to follow. I am also grateful for all the motivation they have given me throughout this journey.

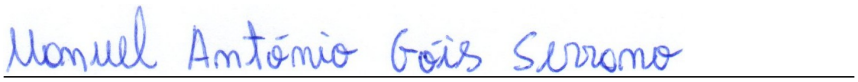
Manuel Serrano

Declaration of honour

I declare that this dissertation is my own work and has not previously been submitted or assessed at this or any other institution. References to other authors (statements, ideas, thoughts), as well as the use of published works, strictly comply with the rules of attribution and are duly identified in the text and in the bibliographic references, in accordance with the applicable referencing standards. I am aware that the practice of plagiarism or self-plagiarism constitutes an academic offence, as established in the U.Porto Code of Ethics.

I also declare that I have used Artificial Intelligence tools for linguistic and stylistic revision of the text and to support bibliographic research in completing this dissertation, and that this use and its results are duly noted and have been carefully checked for errors, inaccuracies, unfounded attributions or other forms of bias in content and arguments, as well as determining authorship.

Manuel António Góis Serrano



Contents

List of Figures

List of Tables

List of Acronyms

ACF Autocorrelation Function

ADI Average Demand Interval

AI Artificial Intelligence

ARIMA Autoregressive Integrated Moving Average

BiLSTM Bidirectional LSTM

CNN Convolutional Neural Network

CV^2 Squared Coefficient of Variation of Non-zero Demand Sizes

DL Deep Learning

DRFAM Direct and Recursive Forecast Averaging Method via Partial Pooling

ELU Exponential Linear Unit

ES Exponential Smoothing

ETS Error, Trend, Seasonality

FLOP Floating Point Operation

GMLP Global Multi-Layer Perceptron

GRF Global Random Forest

GRN Gated Residual Network

IQR Interquartile Range

LightGBM Light Gradient Boosting Machine

LSTM Long Short-Term Memory

MAE Mean Absolute Error

MASE Mean Absolute Scaled Error

ML Machine Learning

MLP Multi-Layer Perceptron

MSE Mean Square Error

PACF Partial Autocorrelation Function

PICP Prediction Interval Coverage Probability

ReLU Rectified Linear Unit

RF Random Forest

RMSSE Root Mean Squared Scaled Error

RNN Recurrent Neural Network

SARIMA Seasonal Autoregressive Integrated Moving Average

SGD Stochastic Gradient Descent

TCN Temporal Convolutional Network

TPE Tree-structured Parzen Estimator

TFT Temporal Fusion Transformer

US United States

VSN Variable Selection Network

WRMSSE Weighted Root Mean Squared Scaled Error

Chapter 1

Introduction

This chapter introduces the dissertation. Section 1.1 presents the food retail context and the main challenges of demand forecasting, including volatility and product heterogeneity. Section 1.2 sets out the motivation for this work and highlights the practical relevance of evaluating advanced Deep Learning (DL) methods in real-world settings. Section 1.3 formulates the problem addressed, Section 1.4 states the research questions that structure the remainder of the dissertation, and Section 1.5 describes its overall organisation.

1.1 Context

The food retail sector is a commercial activity of great relevance, not only in the food dimension but also economically. It represents a significant share of the global market, with the total turnover of european food retailers alone reaching €3.5 trillion in 2018 [73]. Since its origins, the sector has evolved continuously and currently relies on Artificial Intelligence (AI)-based systems to improve demand forecasting, that is, the estimation of the quantities of products that consumers will purchase over a given period, and supply chain management [11]. These systems employ Machine Learning (ML) algorithms to analyse historical data and exogenous variables in order to estimate future demand with greater accuracy [11].

However, forecasting demand in food retail remains far from trivial. This complexity arises from the high diversity of products commercialised. Moreover, inaccurate forecasts can lead to significant consequences, such as stockouts or food waste [4]. This sector simultaneously deals with fresh and long-life products, each with distinct characteristics and consumption patterns. Fresh products, in particular, have short shelf lives, meaning that incorrect forecasts can quickly lead to significant losses. Non-perishable products, although less sensitive over time, can also generate storage costs if quantities are not correctly estimated. Thus, the diversity of the food retail portfolio substantially increases the complexity of demand forecasting [3].

Additionally, demand in food retail exhibits high volatility due to multiple factors such as seasonality, promotions, weather conditions, and market trends [1]. In highly promotional environments, this volatility can lead to demand peaks up to 60 times higher than the baseline, followed

by abrupt post-promotion drops [1], creating discontinuities that hinder modelling.

Given these challenges, demand forecasting accuracy becomes critical not only for operational efficiency but also for the competitiveness and sustainability of the sector. Forecasting errors propagate along the supply chain, amplifying inventory costs, compromising customer service levels [76], and contributing to food waste. On the other hand, even modest improvements in predictive accuracy can result in significant economic gains when scaled to the operational level typical of the food retail industry [19]. In this context, it becomes essential to understand which approaches are most effective in dealing with the volatility and complexity that characterise this sector.

1.2 Motivation

Although there is extensive literature on demand forecasting, many studies remain confined to public aggregated datasets or laboratory scenarios, without demonstrating practical feasibility in real-world food retail contexts [19]. This disconnect is linked to challenges such as the integration of models with enterprise systems, as well as their scalability and interpretability in business environments [61].

The urgency of addressing these challenges is evident in the economic and environmental impacts of the sector. The contribution of retail to food waste varies significantly across regions and periods, with estimates ranging from 5–12%. This variability reflects methodological limitations in the quantification of the problem [78]. Inadequate demand forecasting systems have been identified as a cause of waste within the food supply chain [62], with research involving industry professionals revealing that data-driven forecasting methods have strong potential to reduce waste in perishable products [4]. However, the transition of these capabilities from laboratory research to operational-scale practice remains a recognised challenge.

Considering these challenges, it becomes imperative to develop and validate approaches that not only demonstrate predictive accuracy in controlled environments but can also be effectively implemented at operational scale. This dissertation, benefiting from access to large-scale real-world data, aims to investigate the potential of advanced DL techniques under non-controlled conditions. The goal is to contribute empirical evidence on whether these models can deliver practical value in a sector where forecasting accuracy has direct consequences for both economic performance and environmental sustainability.

1.3 Problem

There is a need to validate advanced techniques in real operational contexts. In response, this dissertation focuses on applying and evaluating DL models for demand forecasting in the food retail sector. The empirical work is conducted using real operational data. The analysis focuses on architectures such as Long Short-Term Memory (LSTM), Temporal Fusion Transformer (TFT), and a variant of DeepAR.

The central problem consists of assessing whether these models, when trained on large-scale real-world data, can produce more accurate forecasts than traditional approaches. At the same time, the study aims to determine whether they can effectively handle the heterogeneity of products that exhibit distinct demand patterns. Furthermore, it investigates whether these models maintain robust performance under conditions of high volatility, such as promotional periods or seasonal events, and whether they can scale computationally to process thousands of time series simultaneously.

This problem lies at the intersection of methodological research and practical application, encompassing questions of predictive accuracy, computational scalability, and robustness to heterogeneous demand patterns. The use of large-scale real-world data enables a more comprehensive empirical evaluation than studies confined to aggregated public datasets.

1.4 Research Questions

Based on the challenges identified and the problem formulated, this dissertation seeks to address a set of research questions that guide the analysis and validation of the proposed approaches.

RQ1: To what extent can DL models improve the accuracy of demand forecasting in the food retail sector when trained on real operational data?
This question aims to evaluate the predictive accuracy of the proposed models on real data, determining whether DL techniques offer significant improvements compared to classical statistical and traditional ML methods, using error metrics suited to the food retail context.

RQ2: How do these models compare in terms of scalability when trained on an increasing number of stores?
This question investigates how the computational cost and forecasting accuracy of each DL model evolve as the number of stores in the training corpus increases, determining whether the models can scale efficiently to larger retail portfolios without disproportionate degradation in either dimension.

RQ3: How robust are these models when exposed to volatile and complex demand patterns in the food retail sector?
This question examines whether the predictive performance of each DL model remains stable when confronted with demand patterns of varying complexity and irregularity across a large-scale food retail portfolio.

1.5 Document Structure

This dissertation is organised into seven chapters. Chapter 2 introduces the foundational concepts in time series analysis and deep learning that underpin the methodological choices made throughout the study. Chapter 3 reviews the state of the art in demand forecasting for food retail, covering

the main technical challenges, the DL architectures most relevant to this setting, and the considerations that motivate the experimental design. Chapter 4 formalises the forecasting problem and documents the modelling decisions shared across all three architectures. Chapter 5 describes the dataset, feature engineering, experimental pipeline, evaluation metrics, and computational infrastructure used throughout the empirical work. Chapter 6 presents the results of the three research questions, covering predictive accuracy, scalability, and robustness to heterogeneous demand patterns. Chapter 7 draws conclusions from the empirical findings and outlines directions for future work.

Chapter 2

Background

This chapter introduces the foundational concepts that underpin the work presented in this dissertation. Section 2.1 covers the fundamental properties of time series, including their structural components, stationarity, autocorrelation, and the evaluation and forecasting strategies relevant to sequential data. Section 2.2 introduces the core principles of neural networks and DL, covering the building blocks of modern architectures, the mechanisms by which they are trained, and the techniques used to improve their generalisation. The concepts introduced here are assumed as prior knowledge throughout the remainder of the study.

2.1 Time Series

A time series is a sequence of observations indexed in chronological order, formally defined as $\{y_t\}_{t=1}^{T}$, where $y_t \in \mathbb{R}$ denotes the value of the variable of interest at time t and T is the total number of observations. The defining characteristic of a time series is that the ordering of observations is not arbitrary: the temporal index carries information about the data-generating process, and the value at any given time step may depend on values observed at previous steps. This temporal dependence distinguishes time series analysis from cross-sectional statistical methods, where observations are assumed to be independent [6].

Time series arise naturally in a wide range of domains, from economics and finance to meteorology and retail operations [30]. In the context of demand forecasting, the variable of interest is typically the quantity of a product sold over successive time periods, and the objective is to exploit the structure present in the historical sequence to produce reliable estimates of future values [76].

2.1.1 Univariate and Multivariate Time Series

A univariate time series consists of a single sequence of observations indexed in time, $\{y_t\}_{t=1}^{T}$, where the goal is to model the evolution of one variable over time. In the forecasting setting, predictions of future values rely exclusively on the past history of that same variable.

A multivariate time series extends this to a collection of M variables observed simultaneously at each time step, represented as a sequence of vectors $\{y_t\}_{t=1}^{T}$ where $y_t \in \mathbb{R}^M$. In this setting,

the observation at each time step carries information about multiple variables, and the forecasting model can exploit relationships across variables to improve predictions. A special case arises when the goal is to forecast only a subset of the M variables whilst using the remaining ones purely as inputs, a formulation commonly adopted when external signals are available to inform the forecast [44].

2.1.2 Components of a Time Series

The observed values of a time series are commonly understood as the superposition of several underlying components, each capturing a distinct aspect of the data-generating process. The classical decomposition framework expresses the observed series as a combination of four components: trend, seasonality, cyclical variation, and an irregular component [53].

The trend component T_t describes the long-run direction of the series, capturing sustained increases or decreases in the level of the variable over time. In retail demand, trend may reflect gradual shifts in consumer preferences, population growth, or the progressive expansion of a product's presence in the market.

The seasonal component S_t captures periodic fluctuations that recur at fixed and known intervals, such as daily, weekly, or annual cycles. Seasonality in retail is particularly pronounced, with demand for many products exhibiting strong weekly patterns driven by shopping behaviour, as well as annual cycles associated with holidays and promotional calendars [19].

The cyclical component C_t refers to medium-to-long-term oscillations that are not of fixed periodicity, often associated with broader economic cycles. Unlike seasonality, cyclical variation does not follow a rigid calendar schedule and is therefore harder to model explicitly.

The irregular component ε_t, also referred to as the residual or noise component, captures the portion of the observed series that cannot be attributed to any of the systematic components above. It is typically assumed to be a zero-mean random process with finite variance [6].

These components can be combined in two canonical ways. The additive decomposition assumes that the components sum linearly, as shown in Eq. 2.1:

$$y_t = T_t + S_t + C_t + \varepsilon_t \tag{2.1}$$

whilst the multiplicative decomposition assumes that they interact, as defined in Eq. 2.2:

$$y_t = T_t \cdot S_t \cdot C_t \cdot \varepsilon_t \tag{2.2}$$

The choice between the two depends on whether the amplitude of seasonal fluctuations grows proportionally with the level of the series. When seasonal swings scale with the trend, the multiplicative form is more appropriate; when they remain constant in absolute terms, the additive form is preferred [30].

2.1.3 Stationarity

A time series is said to be strictly stationary if its joint probability distribution is invariant to shifts in time, meaning that the statistical properties of the series do not change as a function of when they are measured. In practice, a weaker condition is typically assumed: a series is covariance stationary, or weakly stationary, if its mean and variance are constant over time and its autocovariance depends only on the lag between observations, not on the absolute time index. Formally, a process $\{y_t\}$ is weakly stationary if:

$$\mathbb{E}[y_t] = \mu \quad \forall t \tag{2.3}$$

$$\text{Var}(y_t) = \sigma^2 < \infty \quad \forall t \tag{2.4}$$

$$\text{Cov}(y_t, y_{t-k}) = \gamma(k) \quad \forall t, k \tag{2.5}$$

where μ and σ^2 are constants and $\gamma(k)$ depends only on the lag k.

Stationarity is a foundational assumption of many classical time series models, as it ensures that the patterns estimated from historical data remain valid for future periods [6]. Non-stationary series, which exhibit time-varying means or variances, violate this assumption and can lead to spurious statistical relationships if analysed without appropriate transformation. Common sources of non-stationarity include deterministic trends, which can be removed by differencing or detrending, and heteroscedasticity [6], where the variance of the series changes over time. In retail demand, non-stationarity is pervasive: promotional events, seasonal peaks, and structural shifts in consumer behaviour all introduce time-varying statistical properties that challenge the assumptions of stationary models [19].

2.1.4 Autocorrelation

A central concept in time series analysis is the dependence between observations separated by a given number of time steps. The autocovariance at lag k is defined as shown in Eq. 2.6:

$$\gamma(k) = \text{Cov}(y_t, y_{t-k}) = \mathbb{E}[(y_t - \mu)(y_{t-k} - \mu)] \tag{2.6}$$

Normalising by the variance yields the Autocorrelation Function (ACF), defined in Eq. 2.7:

$$\rho(k) = \frac{\gamma(k)}{\gamma(0)} \tag{2.7}$$

where $\rho(k) \in [-1, 1]$ measures the linear dependence between y_t and y_{t-k}. The ACF provides a compact summary of the temporal memory of a series: a slowly decaying ACF indicates that observations remain correlated over long lags, whilst a rapidly decaying ACF suggests that the series has short memory and recent observations carry most of the predictive information.

A complementary tool is the Partial Autocorrelation Function (PACF), which measures the correlation between y_t and y_{t-k} after removing the linear influence of all intermediate lags $y_{t-1}, \ldots, y_{t-k+1}$. Together, the ACF and PACF are used to identify the order of autoregressive and moving

average components in classical time series models [6], and they provide intuition about the lag structure that forecasting models must capture to produce accurate predictions.

2.1.5 Temporal Evaluation Strategies

A fundamental constraint in time series analysis is that the temporal ordering of observations must be respected throughout the modelling pipeline. Unlike cross-sectional data, where observations are exchangeable and a random partition into training and test sets is appropriate, time series data exhibit temporal dependencies that make random splitting invalid. Assigning future observations to the training set and past observations to the test set would allow the model to learn from information that would not be available at prediction time, a phenomenon known as data leakage [37]. To avoid this, any evaluation protocol must ensure that the model is always trained exclusively on observations that precede the evaluation period in time.

The simplest temporally valid evaluation strategy is a single holdout split, in which the series is divided at a fixed cutoff point into a training segment and a test segment. Whilst straightforward to implement, this approach has a limitation: the estimate of generalisation performance depends entirely on the characteristics of the single test window chosen. If that window happens to coincide with an atypical period, such as an unusually volatile season or a structural shift in demand, the resulting performance estimate may not be representative of the model's behaviour across a broader range of conditions.

Walk-forward validation addresses this limitation by evaluating the model across multiple temporally ordered train-test splits [8, 77]. In the expanding window variant, the training set grows with each fold whilst the test window advances forward in time, so that each fold evaluates the model on a distinct and non-overlapping segment of the series. This produces multiple performance estimates that collectively cover a wider range of temporal conditions.

A complementary concept is the sliding window procedure, used to generate training samples from a long time series. Rather than treating the entire training segment as a single observation, the series is divided into overlapping input-output pairs by advancing a fixed-length window across the available history. Each sample consists of a lookback window of length L, containing the historical observations used as model input, followed by a prediction horizon of length h, containing the future values the model is trained to forecast. The window is advanced by a fixed stride s at each step, so that consecutive samples overlap by $L + h - s$ time steps. The choice of stride controls the trade-off between sample diversity and redundancy: a stride of one maximises the number of training samples but introduces high overlap between consecutive windows, whilst a larger stride reduces redundancy at the cost of fewer samples.

2.1.6 Multi-step Forecasting Strategies

Many practical forecasting tasks require predictions not just one step ahead, but across an extended future horizon of h steps. Given a time series $\{y_t\}_{t=1}^{T}$ and a lookback window of length L, the multi-step forecasting problem consists of predicting the vector $y_{T+1:T+h} = (y_{T+1}, y_{T+2}, \ldots, y_{T+h})$ from

the observed history $y_{T-L+1:T}$ and any available covariates. Several strategies exist for structuring this prediction problem, each making different assumptions about how the h future values are produced [5].

The recursive strategy produces multi-step forecasts by applying a one-step-ahead model repeatedly. At each step k, the model generates a prediction $\hat{y}_{T+k}$, which is then appended to the input window to produce the prediction at step $k+1$, as illustrated in Eq. 2.8:

$$\hat{y}_{T+k} = f(y_{T-L+k:T+k-1}), \quad k = 1, \ldots, h \tag{2.8}$$

The direct strategy instead produces all h future values simultaneously from the observed history, without conditioning on intermediate predictions, as shown in Eq. 2.9:

$$(\hat{y}_{T+1}, \ldots, \hat{y}_{T+h}) = f(y_{T-L+1:T}) \tag{2.9}$$

Here, a single model f produces all h predictions in a single forward pass, with no dependency on previously generated outputs.

Each of these strategies represents a different decomposition of the multi-step forecasting problem.

2.2 Neural Networks and Deep Learning

Artificial neural networks are computational models inspired by the structure of biological neural systems [23]. At their core, they are function approximators: given an input, a neural network applies a sequence of parametric transformations to produce an output, and the parameters are adjusted during training so that the output matches a desired target as closely as possible. The power of neural networks derives from their ability to learn complex, non-linear mappings from data without requiring the modeller to specify the functional form explicitly [40].

2.2.1 The Perceptron and Feedforward Networks

The fundamental building block of a neural network is the neuron, or unit, which computes a weighted sum of its inputs and passes the result through a non-linear function. For a single neuron receiving an input vector $x \in \mathbb{R}^d$, the output is given by Eq. 2.10:

$$z = f(w^\top x + b) \tag{2.10}$$

where $w \in \mathbb{R}^d$ is a vector of learnable weights, $b \in \mathbb{R}$ is a bias term, and $f(\cdot)$ is a non-linear activation function. A single neuron of this form is known as a perceptron, and whilst it can represent linear decision boundaries, it cannot capture non-linear relationships on its own.

Stacking multiple neurons into layers and connecting them sequentially yields a feedforward neural network, also referred to as a Multi-Layer Perceptron (MLP). In such a network, the output

of one layer serves as the input to the next, and the composition of multiple non-linear transformations allows the network to represent increasingly abstract features of the input. A network with N layers computes the function defined in Eq. 2.11:

$$h^{(n)} = f^{(n)}\left(W^{(n)}h^{(n-1)} + b^{(n)}\right), \quad n = 1,\ldots,N \tag{2.11}$$

where $h^{(0)} = x$ is the input, $W^{(n)}$ and $b^{(n)}$ are the weight matrix and bias vector of layer n, and $h^{(N)}$ is the final output. The depth of a network, measured by its number of layers, and its width, measured by the number of units per layer, together determine its representational capacity. Networks with more than one hidden layer are referred to as deep neural networks, and the field concerned with their design and training is known as DL [23].

2.2.2 Activation Functions

The choice of activation function $f(\cdot)$ is a fundamental design decision, as it determines the non-linearities that the network can represent. Without non-linear activations, the composition of linear transformations reduces to a single linear map, collapsing the expressive power of the entire network regardless of its depth.

The sigmoid function, defined as $\sigma(x) = 1/(1+e^{-x})$, maps any real input to the interval $(0,1)$ and was historically the default choice for hidden layers. However, its saturating behaviour for large positive or negative inputs causes gradients to vanish during backpropagation [2], slowing or preventing learning in deep networks.

The Rectified Linear Unit (ReLU), defined as $ReLU(x) = max(0,x)$, addresses this by maintaining a constant gradient for positive inputs whilst zeroing negative ones. Its computational simplicity and resistance to vanishing gradients have made it one of the most used activation functions in modern DL [26]. A smooth approximation of ReLU is the Softplus function [21], defined as $Softplus(x) = \log(1+e^{x})$, which approaches zero asymptotically for large negative inputs and grows linearly for large positive ones. Unlike ReLU, Softplus is differentiable everywhere, which makes it particularly suitable when the output must be strictly positive and gradient continuity is required throughout training.

The Exponential Linear Unit (ELU) [10], defined in Eq. 2.12:

$$\text{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^{x}-1) & \text{if } x \leq 0 \end{cases} \tag{2.12}$$

extends ReLU by producing negative outputs for negative inputs, which helps push the mean activations of each layer closer to zero and can accelerate convergence.

2.2.3 Loss Functions

A loss function is a scalar-valued function that quantifies the discrepancy between the predictions produced by a model and the corresponding target values. Given a set of n observations

$\{(x_i, y_i)\}_{i=1}^{n}$, a loss function $\mathscr{L}(\theta)$ maps the model parameters θ to a non-negative real number that summarises how poorly the model performs on the training data. The smaller the value of $\mathscr{L}(\theta)$, the closer the model's predictions are to the observed targets.

The loss function serves two distinct purposes in the training of neural networks. First, it defines the optimisation objective: training consists of finding the parameter values that minimise $\mathscr{L}(\theta)$ over the training set. Second, it encodes an implicit assumption about the distribution of the target variable, since different loss functions correspond to the negative log-likelihood of different probability distributions [23]. The choice of loss function is therefore a modelling decision that should reflect the statistical properties of the data, ensuring that the training objective is aligned with the underlying data-generating process.

2.2.4 Backpropagation and Optimisation

Training a neural network consists of finding the parameter values that minimise a loss function $\mathscr{L}(\theta)$. This is an optimisation problem in a high-dimensional parameter space, and it is solved iteratively using gradient-based methods.

The gradient of the loss with respect to all parameters is computed efficiently using backpropagation, an algorithm that applies the chain rule of calculus to propagate error signals from the output layer back through the network [67]. For a network with N layers, the gradient with respect to the parameters of layer n is obtained by multiplying the local gradient at that layer by the upstream gradient passed from layer $n+1$, as shown in Eq. 2.13:

$$\frac{\partial \mathscr{L}}{\partial W^{(n)}} = \frac{\partial \mathscr{L}}{\partial h^{(n)}} \cdot \frac{\partial h^{(n)}}{\partial W^{(n)}} \tag{2.13}$$

Once the gradients are available, the parameters are updated using an optimisation algorithm. The simplest such algorithm is Stochastic Gradient Descent (SGD), which updates the parameters in the direction of the negative gradient computed on a randomly sampled mini-batch of training examples, as shown in Eq. 2.14:

$$\theta \leftarrow \theta - \eta \nabla_{\theta} \mathscr{L}(\theta) \tag{2.14}$$

where η is the learning rate, a hyperparameter that controls the step size of each update. Whilst SGD is straightforward, it can exhibit slow convergence and sensitivity to the choice of learning rate. Adaptive optimisers address this by maintaining per-parameter learning rates that are adjusted based on the history of gradients. The Adam optimiser, which is used throughout this study, combines momentum, an exponentially weighted moving average of past gradients, with adaptive

scaling based on the squared gradients [38]:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{2.15}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{2.16}$$

$$\theta_t = \theta_{t-1} - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{2.17}$$

where g_t is the gradient at step t, β_1 and β_2 are decay rates, and $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected estimates of the first and second moments of the gradient. Adam is adopted in DL for its robustness to hyperparameter choices and its fast convergence on a broad range of tasks [66].

2.2.5 Regularisation

A central challenge in training neural networks is overfitting, which occurs when the model learns to reproduce the training data too closely, capturing noise rather than the underlying pattern, and consequently generalises poorly to unseen examples Regularisation refers to a collection of techniques designed to reduce overfitting by constraining the complexity of the learned function.

Weight decay, also known as L2 regularisation, adds a penalty term to the loss function proportional to the squared magnitude of the parameters, as shown in Eq. 2.18:

$$\mathscr{L}_{\text{reg}}(\theta) = \mathscr{L}(\theta) + \lambda \|\theta\|^2 \tag{2.18}$$

where λ controls the strength of the penalty. This discourages the network from assigning large weights to any individual parameter, promoting smoother and more generalisable solutions.

Dropout is a stochastic regularisation technique that operates during training by randomly setting a proportion p of the activations in a given layer to zero at each forward pass [74]. This prevents individual neurons from co-adapting too closely to specific training examples, as no unit can rely on the consistent presence of any other. At inference time, dropout is disabled and the activations are scaled by $1-p$ to preserve the expected magnitude of the outputs. Dropout has proven highly effective across a wide range of architectures and is particularly useful in deep networks where the risk of overfitting grows with model capacity [74].

2.2.6 Embeddings

Many real-world datasets contain categorical variables, such as product identifiers, store locations, or day-of-week labels, that cannot be fed directly into a neural network as raw integer codes. Assigning arbitrary integers to categories implies a spurious ordinal relationship that the network may exploit during training, leading to poorly calibrated representations.

Embeddings address this by mapping each categorical value to a dense vector of continuous values in a lower-dimensional space [24] $\mathbb{R}^E$, where E is the embedding dimension. Formally, given a vocabulary of V distinct categories, an embedding layer is a learnable matrix $E \in \mathbb{R}^{V \times E}$,

and the embedding of category i is the i-th row of this matrix. The embedding vectors are initialised randomly and updated during training alongside the other parameters of the network, allowing the model to learn a representation in which semantically or statistically similar categories are mapped to nearby points in the embedding space.

This learned geometry can capture latent structure that would be invisible to one-hot encodings or raw integer codes. In the context of retail demand forecasting, product and store identifiers embedded in this way allow the model to discover shared patterns across items with similar demand profiles or across stores with similar customer behaviour, without requiring any explicit hand-crafted features to encode these relationships.

Chapter 3

State of the Art

This chapter synthesises the state of the art in demand forecasting for food retail, with particular emphasis on DL approaches for time series. Section 3.1 outlines the key technical challenges in retail demand, including intermittency, sparsity, and non-stationarity. Section 3.2 reviews the main DL architectures for time series forecasting, covering recurrent, attention-based, and autoregressive approaches. Sections 3.3 and 3.4 discuss probabilistic forecasting and uncertainty, as well as scalability considerations and the role of global models in large-scale retail settings. Section 3.5 examines the integration of exogenous variables and the practical constraints associated with their use. Section 3.6 summarises the main conclusions drawn from the reviewed literature.

3.1 Technical Challenges in Retail Forecasting

At store-item level, food retail demand forecasting is technically challenging because the underlying process is neither stable nor dense: observations are often sparse, demand can be intermittent, and the statistical properties of the series may shift over time. These characteristics affect both model fit and evaluation under realistic temporal settings, and they limit the extent to which models can generalise across products. This section summarises the main challenges that motivate the methodological choices discussed in the subsequent sections.

3.1.1 Limitations of Traditional Approaches

The complexity of forecasting in the food retail sector does not stem solely from data volume, but from the intrinsic characteristics of the underlying time series. Operational retail data is often high-dimensional, non-linear, and volatile, which can challenge the theoretical assumptions of traditional statistical models [12, 56].

Historically, approaches such as Autoregressive Integrated Moving Average (ARIMA) or Holt-Winters Exponential Smoothing have remained widely used due to their simplicity and interpretability. However, these parametric models typically rely on stationarity (often achieved through differencing) and on distributional assumptions that are convenient for estimation and inference [16]. In the context of modern food retail, these assumptions may be challenged by complex consumer

behaviour and promotional dynamics [63]. As noted in [71], transforming series to enforce stationarity may attenuate information related to long-term trends and seasonal structure, limiting the ability of these methods to capture the multi-faceted nature of retail demand.

The heterogeneity within product portfolios further complicates forecasting. Retailers must forecast demand for thousands of distinct products simultaneously, ranging from fast-moving goods to intermittent slow movers [70]. Traditional approaches are typically trained independently for each time series, treating each product in isolation [57]. This univariate perspective ignores potential relationships between products and assumes that sufficient historical data exists for each individual series. For products with limited sales history or irregular demand patterns, the resulting forecasts can exhibit high variance and poor out-of-sample performance [9].

Moreover, traditional methods can face difficulties in representing non-linear relationships between demand and its drivers. Linear models cannot adequately capture complex interactions that characterise retail environments [44, 63]. These methodological constraints limit the ability of classical approaches to adapt to the diverse and dynamic nature of food retail demand [12, 16].

3.1.2 Sparsity and Intermittency

A defining characteristic of food retail datasets is the prevalence of intermittent demand patterns, often resulting in "zero-inflated" time series. While aggregated sales figures may appear relatively smooth, store-item series frequently contain extended runs of zero observed sales [57]. This sparsity poses a fundamental challenge for forecasting models, as the abundance of zero values can dominate the training signal and bias standard regression objectives, particularly those minimising Mean Absolute Error (MAE), towards conservative predictions close to zero [60]. The resulting forecasts tend to underestimate demand during periods of non-zero sales [39]. Figure 3.1 illustrates the heavy-tailed concentration of sales across items, in which many products exhibit a very low sales frequency.

Figure 3.1: Long-tailed distribution of item sales frequency (log–log scale), highlighting the prevalence of low-activity items in retail datasets [70].

This sparsity phenomenon is further complicated by the distinction between genuine lack of demand and censored demand. As highlighted in [54], a recorded zero in historical sales data is ambiguous: it may represent a day where no customers wished to purchase the product, or a day

where the product was unavailable due to stockout, rendering the demand unobservable. Training a model on raw sales data without correcting for these stockouts can introduces a structural error, as the model learns to predict low sales during periods where potential demand might have been high. For products with frequent stockouts, this can lead to systematic underestimation of true demand potential [54].

The sparsity of demand data creates specific modelling challenges beyond the censoring problem. For slow-moving items, the limited number of non-zero observations provides limited informative signal, making it difficult to distinguish genuine patterns from noise [9, 60]. Moreover, the preponderance of zeros can obscure temporal dependencies. Models may struggle to learn meaningful dynamics when most observations are identical (zero), leading to forecasts that fail to anticipate the timing and magnitude of actual sales events [33]. This challenge is particularly acute for new products or items with seasonal demand, where the scarcity of relevant historical data exacerbates the sparsity problem [13].

3.1.3 Volatility and Non-Stationarity

Retail time series are non-stationary, manifesting through abrupt changes in level and variance that may be interpreted as structural breaks in the data [49]. Rather than purely random fluctuations, volatility in retail demand is frequently associated with observable events, such as promotional campaigns or "calendric special days", which can temporarily alter the underlying demand regime [63]. During these periods, the distribution of sales can diverge from recent history, generating peaks that are orders of magnitude higher than the baseline [49]. In this context, this effect is particularly pronounced in perishable categories, where price sensitivity contributes to volatility relative to more stable product lines [16].

This instability also exhibits substantial heterogeneity across store locations, seasons, and product categories [35]. As relationships between inputs and demand can evolve over time. Models trained on relatively stable periods may degrade during high-volatility seasons due to shifts in the underlying data-generating process [33]. The dynamic nature of retail environments means that patterns identified in historical data may not remain valid: promotional strategies change, competitive dynamics shift, and consumer preferences evolve [58, 79]. Moreover, the interaction between volatility and other data characteristics amplifies forecasting difficulty. For instance, promotional events not only create demand spikes but can also induce post-promotion dips as customers anticipate future offers or deplete their inventory [16, 29]. These temporal dependencies and non-linear effects challenge the assumption of stable statistical relationships that underlies many traditional forecasting approaches [44, 71].

3.2 Deep Learning Architectures for Time Series

The modelling response to these challenges is to use architectures that can represent temporal context more flexibly and generate forecasts over multiple horizons. This section describes the main

DL architectural choices for time series forecasting, distinguishing between model architectures that encode temporal dependencies and forecasting strategies that produce multi-step outputs.

3.2.1 Recurrent Neural Networks and Hybrid Architectures

Early DL approaches to modelling the sequential nature of demand forecasting commonly relied on Recurrent Neural Networks (RNNs). Unlike feed-forward networks, RNNs process data sequences by maintaining an internal memory state, allowing previous inputs to influence current outputs [28]. However, standard RNNs struggle with learning dependencies over long sequences due to the vanishing gradient problem [12]. To address this, the LSTM architecture introduced a gating mechanism that regulates the flow of information, enabling the model to retain relevant long-term patterns while discarding noise [27]. Empirical studies in retail contexts indicate that LSTMs often outperform traditional statistical baselines like Seasonal Autoregressive Integrated Moving Average (SARIMA) in categories with complex seasonality [15]. However, some research notes that this advantage may diminish in highly volatile product categories where statistical methods remain competitive [16].

To further enhance the capacity of recurrent models to capture temporal context, researchers have explored Bidirectional LSTMs (BiLSTMs). While standard LSTMs process data only from the past to the future, BiLSTMs process the sequence in both directions simultaneously. This architecture allows the network to learn from both forward and backward dependencies within the observed input window, which has been found to be particularly effective in capturing non-linear demand fluctuations compared to unidirectional models [34, 55].

A significant evolution in this domain involves architectural hybridisation, specifically the combination of Convolutional Neural Networks (CNNs) with LSTMs. In this sequential design, the CNNs layers function as feature extractors, applying filters to raw input data to identify local patterns or reduce noise before passing the refined features to the LSTM for temporal modelling. Research suggests that this CNN-LSTM approach can capture salient patterns more effectively than LSTMs alone, particularly in datasets characterized by high dimensionality and noise [59, 68].

Beyond feature extraction, hybrid architectures have also been designed to decouple linear and non-linear patterns. For instance, sequential frameworks have been proposed where an LSTM models the primary temporal trend, while a secondary algorithm, such as Random Forest (RF), is trained specifically on the residuals to capture variance driven by external factors that the recurrent network missed [63]. Similarly, to address the specific challenge of intermittent demand in e-commerce, hybrids have been developed that switch between LSTMs and classical parametric methods based on product categorisation, ensuring that the model architecture aligns with the underlying data distribution [60]. Overall, these studies reflect a shift from using pure recurrent networks towards composite architectures designed to mitigate specific structural weaknesses.

3.2.2 Attention Mechanisms and Transformers

Despite their suitability for sequential data, recurrent networks rely on inherently sequential computation, which limits training efficiency and the capture of long-range dependencies [49]. To overcome these barriers, the DL paradigm evolved towards self-attention mechanisms, which process the entire temporal sequence simultaneously. However, unlike RNNs, the standard Transformer architecture lacks an inherent understanding of order. To address this, time series adaptations employ positional encodings, such as the learnable Time2Vec representations used in [43], to inject temporal context, capturing periodic and non-periodic patterns inherent in retail seasonality.

A commonly noted structural advantage of this family is the shift from recursive decoding to direct multi-horizon forecasting. Traditional RNNs typically generate predictions step-by-step, leading to error accumulation over long horizons [83]. In contrast, attention-based architectures employ decoders that generate forecasts for multiple future time steps simultaneously. This non-autoregressive strategy preserves the integrity of the prediction window, which can improve robustness over longer planning horizons, as demonstrated in recent comparative studies against LSTMs [82].

Despite these advantages, the canonical Transformer incurs quadratic computational complexity ($O(L^2)$) regarding sequence length, which can limit practical applicability for high-frequency data [17]. This limitation has driven the development of efficient variants tailored for long-term forecasting. For instance, the Autoformer [81] replaces point-wise self-attention with an auto-correlation mechanism, reducing complexity to $O(L \log L)$ while integrating a series decomposition block deep within the architecture to progressively separate trend and seasonality. Similarly, sparse-attention models like the Informer have been explored to handle long sequences, although recent research [7] suggests that in retail contexts, the inclusion of explanatory variables may outweigh the benefits of complex sparse mechanisms.

In the specific context of heterogeneous retail data, the TFT [44] is widely used as a benchmark architecture. Unlike standard Transformers that treat all features uniformly, the TFT is purpose-built to align multi-modal data. It structurally distinguishes between static covariates (time-invariant attributes), historical inputs (observed only in the past), and known future inputs (deterministically known variables). Through specialised components like Variable Selection Networks (VSNs) and Gated Residual Networks (GRNs), the model dynamically suppresses irrelevant features and aligns these distinct input streams without manual engineering [44, 71].

Beyond predictive accuracy, attention mechanisms offer a distinct advantage in interpretability. By examining attention weights, stakeholders can gain indications of which past time steps or specific features contributed most to a prediction, providing a level of transparency often lacking in black-box models [18, 44]. Nevertheless, given the computational cost of attention layers, recent research has proposed lighter alternatives such as TSMixer [14]. This approach attempts to replicate the mixing of temporal and feature information using MLPs alone, questioning whether complex attention mechanisms are necessary for all forecasting tasks.

3.2.3 Autoregressive Neural Forecasting

Autoregressive neural forecasting follows a recursive formulation, where the prediction at the next time step conditions on values observed or predicted in previous steps. This paradigm allows the model to capture complex temporal dynamics and update its internal representation over the forecast horizon. However, a central limitation of this recursive approach is the phenomenon of error accumulation. Since the model consumes its own predictions as inputs for subsequent steps during inference, slight deviations in the early stages can propagate and accumulate [83]. This contrasts with the direct multi-horizon approaches discussed in the previous section, which predict all future steps jointly to avoid this recursive degradation [69, 82].

To mitigate these stability issues, specific training strategies are employed to bridge the gap between learning and deployment. The standard approach, known as teacher forcing, conditions on ground-truth observations during training, whereas the model must rely on its own generated predictions during the inference phase. This discrepancy, often termed exposure bias, can be detrimental in volatile retail environments. Recent research has addressed this by introducing rolled training strategies, where the model is trained by conditioning on its own generated predictions rather than exclusively on ground truth, aligning the training process more closely with the inference mechanics to improve robustness against noise [33].

This autoregressive logic can be implemented in different ways, including encoder–decoder (Sequence-to-Sequence) formulations that condition on a historical context window and generate future values step-by-step [55]. Separately, a prominent example of autoregressive neural forecasting is the DeepAR architecture [70], which uses recurrent networks (typically LSTMs) trained across multiple time series to learn shared patterns.

Recent literature indicates that the evolution of the DeepAR architecture has been driven by three distinct operational objectives: improving stability in intermittent data, enhancing computational efficiency, and increasing sensitivity to structural breaks. Regarding stability, in [33] proposed the Rolled DeepAR, which modifies the recurrence mechanism to train on generated samples, reducing error propagation in zero-inflated retail series. To address computational efficiency, variants proposed in [83] replace the initial recurrent layers with Temporal Convolutional Networks (TCNs), accelerating feature extraction before the autoregressive stage. Finally, to improve sensitivity to turning points in volatile markets, variants such as DeepARA integrate attention modules within the autoregressive structure. However, empirical results suggest that this added complexity does not always yield gains in sectors with specific noise characteristics like food and drink [42].

In the specific context of retail, autoregressive neural forecasting is particularly useful for handling the granular nature of store-item data. Unlike direct mapping models that generate a fixed vector of predictions, autoregressive models are inherently designed to model the transition of state from one time step to the next. This sequential nature enables flexible integration of time-varying covariates at each step of the decoding process. As a result, the forecast trajectory can adjust dynamically to changing inputs throughout the horizon, without requiring the rigid input

structures often associated with non-recursive models [57, 70].

3.3 Probabilistic Forecasting and Uncertainty

In the volatile environment of food retail, traditional point forecasts, which provide a single point estimate for future demand, are often insufficient for optimal decision-making. Relying solely on the mean or median fails to capture the inherent stochasticity of sales, potentially leading to poor inventory management in scenarios of high variability [45]. To address this, probabilistic forecasting is adopted, aiming to estimate the full probability distribution of future demand [70]. This paradigm enables uncertainty to be decomposed into two categories: aleatoric uncertainty, which arises from the intrinsic noise and randomness in the data generation process, and epistemic uncertainty, which stems from the model's own lack of knowledge regarding the underlying physics of the system or suboptimal parameterisation [22].

Probabilistic forecasting is not a distinct model architecture but rather a forecasting objective that can be applied to various DL backbones. To represent uncertainty mathematically, models typically employ one of two dominant strategies: parametric distributional likelihoods or non-parametric quantile regression. The parametric approach, exemplified by the standard DeepAR framework, assumes that the data follows a known distribution (e.g., Gaussian or Negative Binomial) and trains the network to predict the distribution's shape parameters for each time step [45, 70]. Conversely, architectures like the TFT can be adapted to various forecasting objectives but are frequently configured in probabilistic settings to minimize a quantile loss function. This non-parametric strategy allows the model to estimate specific percentiles of the demand distribution (e.g., the 10th, 50th, and 90th percentiles) without making rigid assumptions about the underlying distribution form, offering greater flexibility when the data distribution is skewed or multi-modal [18, 44].

The selection of the appropriate probabilistic formulation is influenced by the structural characteristics of retail data, particularly intermittency and censoring. Retail demand, especially for perishable products, is often characterised by excess zeros and highly skewed, episodic counts, which challenge standard Gaussian error assumptions. To mitigate this, recent studies have successfully applied specialised distributions, such as the Tweedie distribution [33] or the Negative Binomial [70], which are statistically designed to handle non-negative count data with high variance. Building on the censoring effects discussed earlier [54], probabilistic formulations provide a principled way to represent the resulting ambiguity in observed sales, focusing on the latent demand distribution [70] rather than treating recorded zeros as direct observations of demand [54].

Evaluating these models requires metrics that assess two distinct properties: calibration and sharpness. Calibration measures the reliability of the predicted intervals by ensuring, for instance, that a 90% prediction interval contains the true observation 90% of the time, a property often quantified using the Prediction Interval Coverage Probability (PICP) [41]. However, a wide interval is easily calibrated but provides insufficient precision for operational decisions. Therefore,

complementary metrics are required to assess sharpness, such as the Winkler Score, which penalises intervals for being unnecessarily wide [65]. Coherence can be assessed by checking for "quantile crossing", defined as a phenomenon in which a lower percentile mathematically exceeds a higher one, and detected using the Constraint Score [41].

From an operational perspective, the transition from point to probabilistic forecasting supports more risk-aware supply chain decisions. By quantifying the uncertainty associated with each item, retailers can move beyond fixed inventory policies towards adaptive service level objectives. Conceptually, this allows decision-makers to trade off the risk of overstocking (waste) against the risk of understocking (lost sales) in line with the specific cost profile of each product [54, 70].

3.4 Scalability and Global Models

Scalability in retail forecasting is defined not merely by the volume of data processed, but by the capacity of forecasting systems to learn efficiently across vast product portfolios and deliver consistent predictions at scale [25, 32]. This requirement becomes important when demand must be forecast across many related series, including items with limited historical signal. In such cases, transferring information across series is essential to avoid reliance on isolated, series-specific models trained independently [35, 70].

These scalability requirements have driven a shift from local models, trained independently per time series, to global models that learn shared parameters across large collections of related series [50, 57]. By pooling information across items, global models can transfer statistical strength from high-volume products to those with limited history. This cross-series learning mitigates the data sparsity that limit traditional univariate approaches in large-scale settings [35, 57].

Implementing this transfer effectively requires mechanisms that balance shared learning with the heterogeneity of retail demand. Common strategies include cross-learning and pooling approaches that structure learning across series. Partial pooling refines this approach by sharing parameters within hierarchy levels (e.g., department or category) while retaining sufficient flexibility to capture subgroup-specific behaviour [57]. Evidence from the M5 dataset indicates that partial pooling can provide an effective balance between generalisation and specificity, outperforming both local baselines and fully pooled global models [57]. More adaptive frameworks further combine pooled and series-specific components, selecting or weighting models according to the characteristics of each series [50, 60].

Beyond cross-series transfer, additional structural constraints arise from the hierarchical organisation of retail data. Sales can be aggregated at multiple levels (item, store, region, total company), and a coherent forecasting system should ensure that granular forecasts are consistent with aggregates [14, 57]. Classical hierarchical forecasting typically relies on post-hoc reconciliation methods that adjust forecasts to ensure consistency across hierarchy levels. In contrast, some modern DL approaches adopt bottom-up strategies in which forecasts are produced at the most granular level and subsequently aggregated [9]. Empirical evidence suggests that this approach

can improve accuracy relative to modelling only at higher aggregation levels, as it preserves fine-grained patterns that would otherwise be smoothed out [20, 57].

Deploying global models at scale introduces practical constraints on both training and inference. These constraints are particularly relevant when forecasts must be generated frequently across large collections of series under limited computational resources [47, 80]. Efficiency-oriented designs and deployment settings have been explored to reduce memory and latency overheads while retaining competitive accuracy [80, 83]. This includes lightweight architectural alternatives that replace attention mechanisms with simpler mixing operations, sacrificing some representational capacity in exchange for substantially lower memory and latency overhead [84]. These considerations highlight that scalability is shaped not only by modelling choices, but also by the computational envelope within which forecasting systems must operate [20].

The selection of an appropriate scalable modelling strategy involves balancing the benefits of cross-series learning with operational feasibility. Global models provide a foundation for large-scale forecasting. However, effective deployment depends on selecting appropriate pooling strategies and ensuring hierarchical coherence, so that computational resources translate into measurable gains in forecasting accuracy.

3.5 Integration of Exogenous Variables

Relying exclusively on historical sales patterns is often insufficient to capture the drivers of demand in retail environments. Forecasting models therefore incorporate exogenous variables, i.e., external signals that influence consumer behaviour but are not generated by the demand series itself [7]. In practice, these variables commonly include calendar effects (e.g., holidays and day-of-week patterns), commercial actions (e.g., price changes and promotions), environmental conditions, and static descriptors such as product or store attributes [16, 54, 64]. Incorporating contextual information can help models anticipate externally driven shifts, rather than attributing them to unexplained variation in sales [63, 64].

The effective use of exogenous variables requires careful consideration of temporal availability and dynamics. A key distinction is between known future inputs, such as calendar information or planned promotion schedules, and observed-only inputs, which are only available up to the prediction time and may require lagging or external forecasting to be used for future horizons [7, 18, 44]. This distinction matters because it determines what information is legitimately available across the forecast window and what introduces additional uncertainty [7, 18, 44]. Exogenous variables can also be characterised as static (e.g., store location, product category) or dynamic (e.g., price or promotional status), and this affects how they can be structured and aligned with the target series as the horizon extends [64, 65]. These temporal properties shape how information is represented in forecasting systems and what signals remain available under realistic operational constraints [69].

Several methodological considerations arise when integrating exogenous information into forecasting models. Feature selection is important when many candidate variables are available, since

not all external signals contribute meaningful predictive value [46]. The representation is equally relevant for high-cardinality categorical variables (e.g., product and store identifiers), where appropriate encoding is needed to avoid excessively sparse input spaces while still capturing latent similarity [64]. Temporal alignment introduces additional complexity, particularly when exogenous signals are recorded at different frequencies or exhibit lagged effects that require preprocessing to ensure synchronisation with the target series [65, 79]. Practical data issues also matter. Missing or delayed exogenous data can require imputation or robust handling to avoid inconsistencies between training and deployment [72]. Redundancy and multicollinearity among correlated signals can further degrade stability without improving performance [65].

In retail settings, exogenous influences often act through non-linear and time-dependent patterns [16]. For example, promotional effects may include anticipatory behaviour before an event and demand dips afterwards, which are not well captured by simple additive assumptions [29, 44]. Interactions between variables further complicate attribution. Calendar events may coincide with promotions, and the combined effect can differ from the sum of individual contributions [65]. The predictive value of a covariate depends on its signal-to-noise ratio and on the consistency of its relationship with demand across products and stores [64, 65]. These challenges are amplified in settings with high product diversity and frequent external interventions, where large collections of series must be forecast under heterogeneous exogenous influence [56, 57].

Empirical evidence from retail forecasting studies suggests that exogenous variables can improve forecasting performance, although effects are typically context-dependent [7, 16, 29, 64]. Calendar and promotional variables often provide measurable gains across multiple settings, whereas other signals tend to be category-specific and sensitive to data quality and resolution [69]. Some studies also indicate that the magnitude of improvement diminishes as more variables are added beyond a critical subset [65]. This reinforces the importance of selective integration rather than exhaustive feature inclusion. From a deployment perspective, exogenous variables introduce requirements that extend beyond model design. Signals must be reliably available at inference time and consistent with the training pipeline, which can be challenging for variables subject to reporting delays or dependence on external data sources [44, 48]. Relationships between exogenous drivers and demand can also shift over time, which motivates monitoring and periodic retraining in operational environments [29]. These considerations highlight that effective integration depends on both methodological choices and data infrastructure [46, 56].

3.6 Summary

The literature reviewed in this chapter converges on a consistent point: model capacity alone is not decisive. Performance is often determined by how well the modelling choices align with (i) the sparsity and distributional form of sales, (ii) the stability of demand-generating mechanisms over time, and (iii) what information is available at prediction time.

A first line of work addresses representational limitations of traditional statistical methods and early neural approaches. Classical local models remain attractive for their simplicity and interpretability. But they typically struggle to scale across heterogeneous item populations and to capture non-linear effects under distribution shift. Deep architectures alleviate some of these limitations by learning richer temporal representations and by enabling parameter sharing across series. However, this flexibility introduces new sensitivities, particularly to leakage, mis-specified input availability, and instability when training objectives are not aligned with the inference regime.

Across DL architectures, the reviewed evidence suggests that gains are typically realised through structural adaptations rather than wholesale complexity. Recurrent models and their hybrids improve temporal modelling under limited context windows, but can suffer from recursive error accumulation and difficulties in exploiting long-range structure. Attention-based models mitigate some of these issues through more direct access to historical context and direct multi-horizon decoding, yet their computational profile can be a practical constraint. As a result, the literature increasingly positions architectural selection as an exercise in trading off representational power against robustness and operational feasibility, rather than universally favouring complexity.

Probabilistic forecasting reframes the problem from predicting a single trajectory to representing uncertainty in a way that is decision-relevant. This shift is particularly important in retail settings where sparsity, skewness, and censoring make point estimates brittle. Nevertheless, probabilistic outputs are only useful when they are well-calibrated and coherent, and when the evaluation protocol reflects the intended decision context. This introduces an additional methodological burden: selecting likelihoods or quantile objectives consistent with the data-generation process, and validating the resulting distributions with metrics that go beyond average error.

Scalability and global modelling provide a complementary perspective: many practical failures arise not from poor modelling of a single series, but from the inability to generalise across thousands of related products with limited individual history. Global models, pooling strategies, and hierarchical consistency constraints directly target this setting. However, they also expose a recurring tension: aggressive sharing can dilute subgroup-specific dynamics, while overly fine segmentation reduces statistical strength and increases operational complexity. Effective approaches, therefore, tend to sit between these extremes, using mechanisms that control what is shared and where specialisation is retained.

Finally, the integration of exogenous variables is consistently portrayed as necessary but non-trivial. External signals can improve forecast responsiveness and help distinguish systematic effects from noise. At the same time, they amplify risks that are less visible in univariate settings, including temporal misalignment, lagged effects, redundancy, and shifts in the covariate and demand relationship over time. In practice, the value of exogenous information depends on disciplined handling of availability at inference time and on selective inclusion, rather than exhaustive feature accumulation.

Chapter 4

Problem Definition and Approach

This chapter defines the forecasting problem addressed in this dissertation and describes the modelling approach adopted to tackle it. Section 4.1 formalises the forecasting task and establishes the core design constraints that apply across all models. Section 4.2 documents the modelling decisions shared by all three architectures, including the training objective, output activation, and input window length. Section 4.3 describes the concrete implementation of each architecture and the adaptations made relative to the original formulations.

4.1 Problem Formulation

Let $\{y_{i,t}\}_{t=1}^{T}$ denote the daily unit sales of store-item series i, where $i = 1,\ldots,N$ indexes the full portfolio of N store-item pairs. Given a lookback window of length L ending at time T, the forecasting task consists of predicting the vector $\hat{y}_{i,T+1:T+h} = (\hat{y}_{i,T+1},\ldots,\hat{y}_{i,T+h})$ for each series i, where $h = 28$ is the forecast horizon.

Forecasting demand in food retail at store-item level is inherently a large-scale problem. A single retailer may operate hundreds of stores, each carrying thousands of distinct products, producing a portfolio of time series that can scarcely be handled feasibly by a local modelling approach, one series at a time. This practical reality motivates the central design decision of this work: all three architectures evaluated, namely LSTM, TFT, and a non-autoregressive variant of DeepAR, are trained as global models, fitting a single set of parameters jointly across the full collection of store-item series. This allows the models to scale naturally to the size of real retail portfolios, and to leverage patterns shared across series that would otherwise be invisible to a model trained in isolation.

The forecasting task itself is defined at daily granularity, with a prediction horizon of 28 days. This horizon is adopted from the M5 forecasting competition, which provides the dataset underpinning the empirical evaluation. The 28-day window also enables a direct comparison of results against published M5 benchmarks and competition participants, providing an external reference point that goes beyond internal model comparisons.

Each model is designed to produce a forecast in a single forward pass: given a window of historical observations and associated covariates, it outputs a vector of 28 values simultaneously, one per day of the horizon. Each value represents a single point estimate of expected demand, rather than a distribution or prediction interval. This direct multi-step formulation is a deliberate choice, motivated by the well-documented risks of autoregressive decoding in settings with thousands of heterogeneous series, where error accumulation would manifest differently across products and stores and be difficult to control systematically.

The target variable throughout is raw unit sales per store-item-day. This quantity is non-negative, right-skewed, and frequently zero, particularly for slow-moving products and smaller store locations. These distributional properties are not incidental: they are a defining characteristic of demand at this level of granularity, and any modelling choice that ignores them risks producing forecasts that are systematically biased or poorly calibrated. Addressing them requires a set of design decisions that apply uniformly across all three architectures evaluated in this work.

4.2 Shared Modelling Decisions

The first of these decisions concerns the training objective. Standard regression losses such as Mean Square Error (MSE) implicitly assume that the target variable follows a Gaussian distribution, an assumption that does not hold given the distributional properties described above. All three models are therefore trained using the Tweedie deviance loss. The Tweedie deviance is derived from the negative log-likelihood of the Tweedie distribution, a family of distributions that, for a power parameter ρ in the interval $(1,2)$, corresponds to a compound Poisson-Gamma formulation. This regime assigns non-zero probability mass to exact zeros while accommodating the heavy tail of positive values, making it statistically well-suited to the distributional characteristics of retail demand. Minimising the Tweedie deviance during training is equivalent to maximising the likelihood under that distribution, ensuring that the loss function is statistically grounded in the same assumptions that make it suitable for this type of data, without requiring the model to produce probabilistic outputs. The full deviance is defined in Eq. 4.1:

$$D(y,\hat{y}) = 2 \cdot \left(\frac{y^{2-\rho}}{(1-\rho)\cdot(2-\rho)} - \frac{y \cdot \hat{y}^{1-\rho}}{1-\rho} + \frac{\hat{y}^{2-\rho}}{2-\rho} \right) \tag{4.1}$$

where y is the observed sales count and $\hat{y}$ is the model prediction. Since the first term depends only on the observed values and not on the model parameters, it is constant with respect to optimisation and is omitted in the implementation without affecting the training dynamics. The resulting loss function used during training is therefore given in Eq. 4.2:

$$\mathscr{L}(y,\hat{y}) = \left(-\frac{y \cdot \hat{y}^{1-\rho}}{1-\rho} + \frac{\hat{y}^{2-\rho}}{2-\rho} \right) \tag{4.2}$$

The Tweedie deviance requires that predictions be strictly positive. This is enforced by passing the raw model output through a Softplus activation before computing the loss. Unlike a hard

clipping or truncation operation, which would force negative outputs to zero and result in zero gradients during backpropagation, Softplus preserves gradient information throughout training. This smooth behaviour guarantees that outputs are always strictly greater than zero, satisfying the domain requirements of the Tweedie deviance without introducing discontinuities in the optimisation landscape.

The third shared decision concerns the length of the historical input window. All models receive a lookback window of 28 days, matching the length of the forecast horizon. A longer window would provide more historical context but increases memory consumption and training time substantially, which is a meaningful constraint when training globally across thousands of store-item series. A shorter window risks discarding relevant recent history. The 28-day window strikes a practical balance and aligns with the forecasting setup adopted in [33], associated with the third-placed solution in the M5 competition.

4.3 Model Architectures and Adaptations

The three architectures described in this section span a range of approaches to sequence modelling for multi-step forecasting. The LSTM follows a well-established recurrent design. The TFT extends this with structured handling of heterogeneous input types and an attention mechanism. The DeepAR-Direct retains the encoder-decoder recurrent structure of the original DeepAR whilst adopting the direct forecasting strategy shared by all three models. Each sub-section describes the concrete implementation of each architecture and the adaptations made relative to the original formulations.

4.3.1 Long Short-Term Memory

The LSTM architecture implemented in this study is based on the standard formulation presented in [27], adapted for global multi-series forecasting at store-item level. Figure 4.1 illustrates the complete architecture.

The model receives two types of input simultaneously: temporal features and categorical features. The temporal features form a sequence of length L, where each timestep contains F observed values. The categorical features, of which there are C in this implementation, are each mapped to a dense vector of dimension E through a dedicated embedding layer. These embeddings are time-invariant representations: once looked up, each embedding vector is expanded and replicated across all L timesteps, producing a representation of dimension $C \cdot E$ per timestep. The temporal features and the expanded embeddings are then concatenated along the feature dimension, forming a unified input sequence of dimension $F + C \cdot E$ that is fed to the LSTM encoder at each timestep. This design allows the model to condition its temporal reasoning on the identity of each series throughout the entire input window, rather than only at the point of prediction.

The LSTM encoder processes this sequence through a stack of recurrent layers, each maintaining a hidden state of size H. Inter-layer dropout is applied between successive layers when

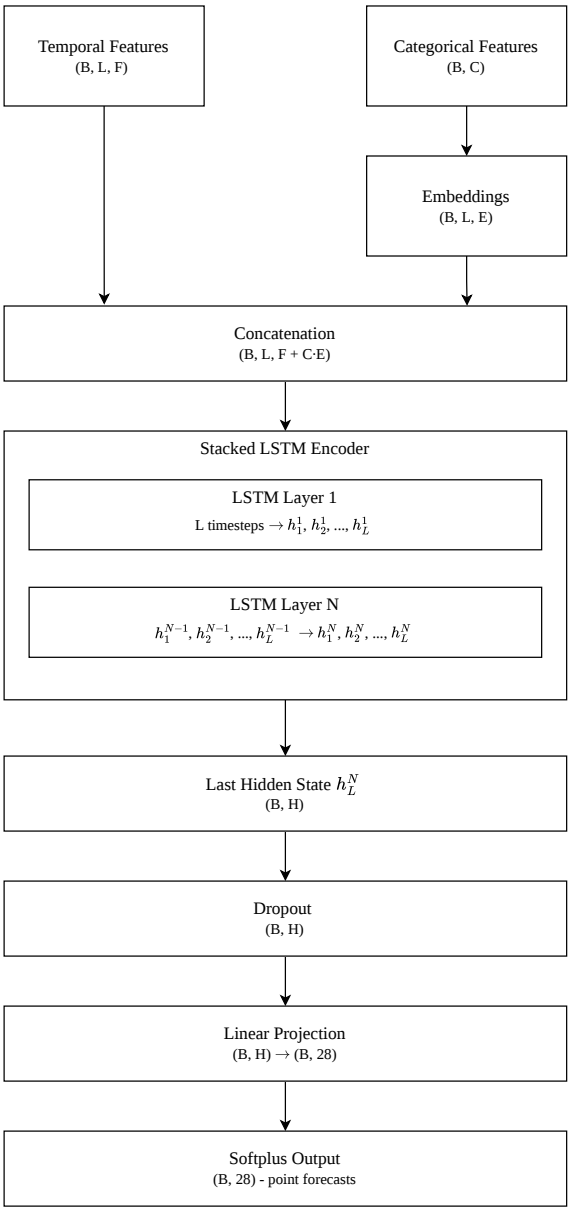


Figure 4.1: Architecture of the global LSTM forecaster with categorical embeddings and direct multi-step output.

more than one layer is used, reducing the risk of overfitting by preventing co-adaptation of features across layers. Only the hidden state produced at the final timestep of the last layer is retained, discarding the cell state and all intermediate hidden states. This is a deliberate choice of simplicity: rather than incorporating an attention mechanism over the full sequence of hidden states, the model relies on the LSTM's inherent ability to summarise the input window into a single fixed-length vector. This keeps the architecture simple and the comparison with the TFT meaningful, as attention-based processing of temporal context is a defining characteristic of that model.

A second dropout operation is then applied to this vector before it is passed to the output stage, providing an additional layer of regularisation against overfitting that is independent of the encoder depth.

The output stage consists of a single linear layer that projects the hidden state vector of dimension H directly to a vector of 28 values, one per day of the forecast horizon. The representational burden is placed entirely on the encoder: if the LSTM has learned a sufficiently rich summary of the input sequence, a single linear transformation is sufficient to produce coherent forecasts across the full horizon. The resulting values are passed through a Softplus activation to enforce strict positivity, as required by the Tweedie training objective.

4.3.2 Temporal Fusion Transformer

The TFT implemented in this study follows closely the architecture proposed in [44], illustrated in Figure 4.2. The three deviations from the original paper concern exclusively the forecasting objective and the output layer: the original quantile loss is replaced by the Tweedie deviance loss, the multi-quantile output head is replaced by a single linear projection, and a Softplus activation is applied to enforce strict positivity, as established in Sections 4.1 and 4.2. The remainder of the architecture is implemented as described in the original paper.

The TFT processes inputs according to the three-stream structure of the original paper: static covariates, represented as learned embeddings of dimension E; observed past inputs over the lookback window; and known future inputs available across the full forecast horizon. Each stream is handled by a dedicated processing pathway before being integrated.

Static inputs enter the model through a Static VSN, whose output serves a purpose that goes beyond simple feature selection: it is the source from which four distinct context vectors are derived, each conditioning a different stage of the temporal processing pipeline. The first, c_s, conditions the variable selection applied to temporal inputs. The second, c_e, conditions the static enrichment layer. The remaining two, c_h and c_c, initialise the hidden and cell states of the LSTM encoder. Through these four vectors, static identity information actively shapes the behaviour of every subsequent component, rather than being injected at a single point.

Temporal inputs, both past and future, are each processed by a dedicated VSN conditioned on c_s. Each scalar feature is first projected to the model's hidden dimension H through a dedicated linear layer before entering its corresponding VSN, which produces a single vector of dimension H per timestep. These selected representations are then processed by an LSTM encoder-decoder, whose purpose is to build a temporally coherent context from the observed history and transfer

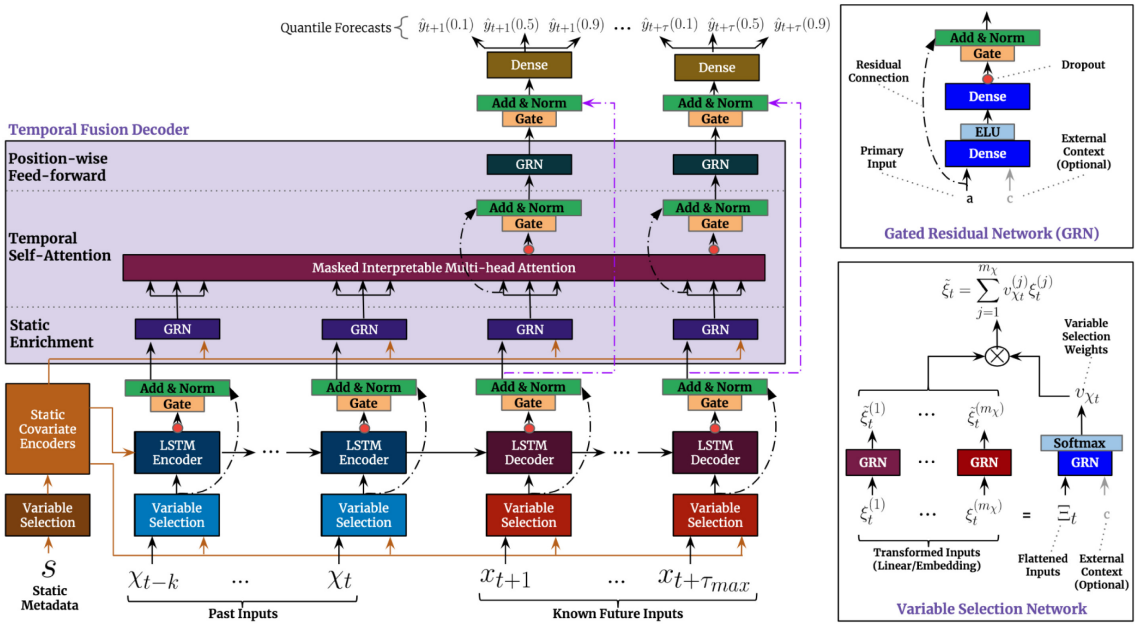


Figure 4.2: Architecture of the TFT [44]. In this study, the multi-quantile output head is replaced by a single linear projection followed by a Softplus activation, and the quantile loss is replaced by the Tweedie deviance loss.

it to the future processing pathway. The encoder processes the past sequence, initialised with c_h and c_c, compressing the historical signal into a set of hidden states that are then used to initialise the decoder. The decoder processes the future sequence from this learned starting point, ensuring that the representations of future timesteps are grounded in the observed past rather than treated independently. A GateAddNorm operation, combining a gated linear unit, a residual addition, and layer normalisation, is applied to both encoder and decoder outputs, with skip connections taken from the VSN outputs prior to the LSTM.

The LSTM outputs then pass through a static enrichment layer, which uses c_e to reinject series-level context into the temporal representations before they reach the attention mechanism. This step ensures that the attention layer does not operate on purely temporal representations, but on ones that already carry awareness of which series is being forecast. The decoder representations then attend to the full concatenated sequence of encoder and decoder states, with a causal mask applied to the decoder portion to prevent each future position from attending to subsequent ones. The resulting representations are then projected to the final point forecasts through the shared output layer common to all three architectures.

4.3.3 DeepAR-Direct

The architecture implemented in this study is a non-autoregressive variant of DeepAR, proposed in [70], adapted to produce direct point forecasts consistent with the shared experimental design established in Section 4.1. The name DeepAR-Direct reflects the central modification: the autoregressive feedback loop of the original decoder is removed, whilst the encoder-decoder LSTM

structure and the core mechanisms of the original paper are preserved. Figure 4.3 illustrates the encoder-decoder structure.

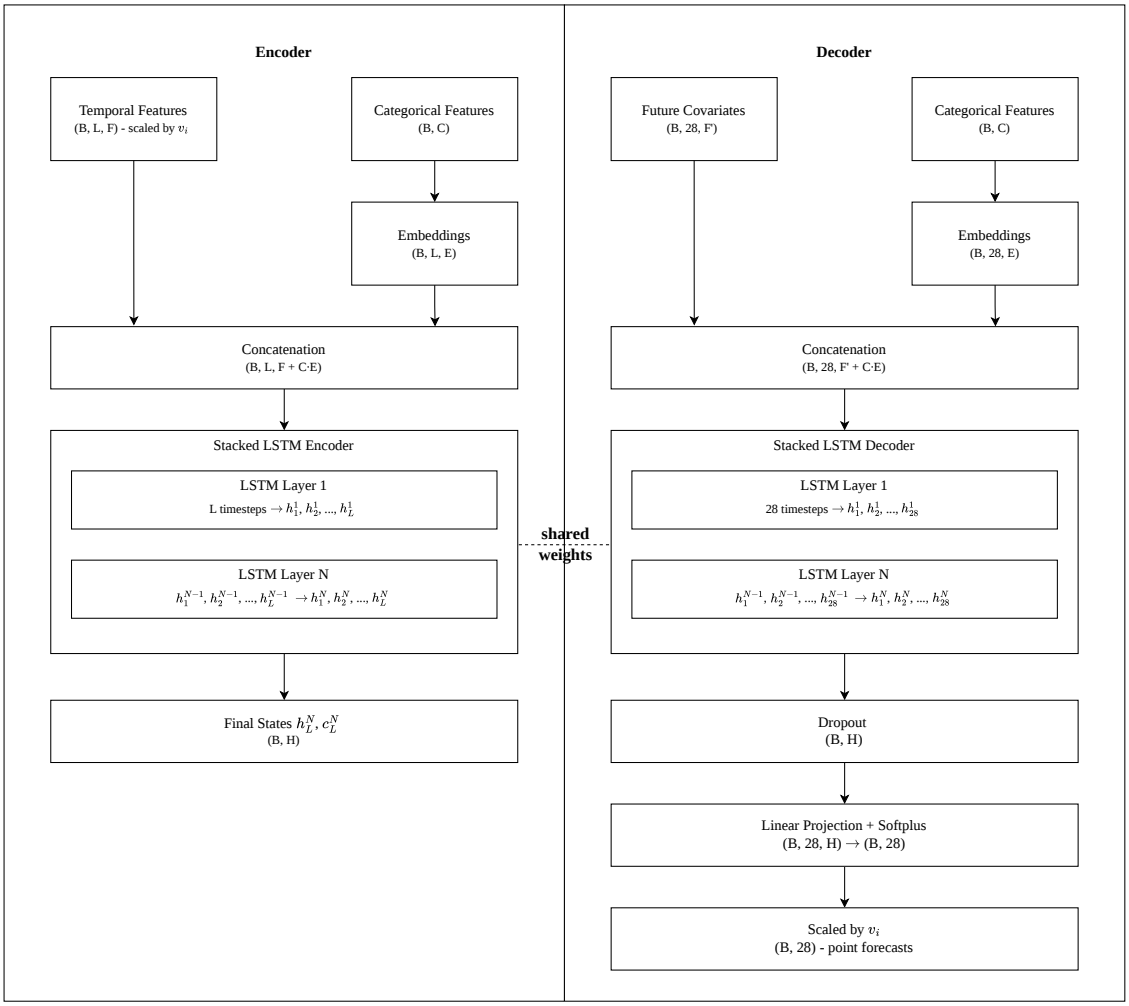


Figure 4.3: Architecture of the DeepAR-Direct forecaster. The decoder is initialised with the final states of the encoder and receives only future covariates and a zero placeholder at each step, replacing the autoregressive feedback of the original DeepAR formulation.

The model follows an encoder-decoder design in which a single LSTM is shared between both stages. The encoder processes the conditioning range, receiving at each timestep the observed features scaled by a series-specific normalisation factor v_i, together with the categorical embeddings expanded across the sequence. This scale factor, defined as $v_i = 1 + mean(z_i)$ over the conditioning range, where z_i denotes the observed sales values of series i, is a mechanism from the original DeepAR paper that addresses the wide range of sales magnitudes across store-item series. By dividing the encoder inputs by v_i, the model operates on a normalised representation that is more stable to learn from, and predictions are subsequently rescaled to the original units at the output stage.

The decoder is initialised with the final hidden and cell states of the encoder, grounding its processing in the observed history before it steps through the 28 future timesteps. At each decoder

step, the LSTM receives only the future-known covariates and the categorical embeddings. The key departure from the original DeepAR formulation is what the decoder does not receive: in the original model, a value sampled from the predicted distribution at step t is fed back as input to step $t+1$, creating an autoregressive dependency across the horizon. In DeepAR-Direct, this feedback is replaced by a zero placeholder. The decoder LSTM still runs sequentially, and its hidden states continue to evolve step by step as future covariates are processed, but no predicted value propagates forward. At each step, the hidden state is passed through dropout for regularisation, projected by a linear layer, and passed through a Softplus activation to produce $\hat{z}_{i,t}$, the point forecast for series i at horizon step t. This value is then rescaled by v_i to recover the forecast in the original scale.

Given the heavily skewed distribution of item velocities that characterises retail portfolios, a naive uniform sampling strategy would underrepresent high-volume series during training. Weighted sampling addresses this by drawing training windows with probability proportional to v_i, ensuring that high-scale series appear more frequently without discarding any series entirely.

Chapter 5

Methodology

This chapter describes the methodology adopted to train and evaluate the three forecasting architectures examined in this study. Section 5.1 characterises the M5 dataset used throughout the empirical evaluation, covering its structure, temporal properties, and demand heterogeneity. Section 5.2 details the feature engineering decisions, documenting which variables are used as model inputs and how they are constructed. Section 5.3 describes the experimental pipeline, including the data partitioning strategy, the hyperparameter optimisation procedure, and the final training protocol. Section 5.4 defines the evaluation metrics used to assess forecast quality. Section 5.5 describes the computational infrastructure on which all experiments were conducted.

5.1 Dataset Description

The M5 Forecasting dataset [52] is a publicly available benchmark derived from real operational data of Walmart, a large United States (US) retailer. It was introduced as part of the M5 Accuracy competition and has since become a standard reference for evaluating forecasting methods at retail scale. Its adoption in this study is motivated by five practical reasons: it contains real operational data at genuine retail scale, rather than simulated or aggregated series; it operates at store-item granularity; it covers a sufficiently long period to capture seasonal variation and structural shifts; it provides a set of published competition results that serve as external reference points for the evaluation; and it exhibits all four demand patterns defined by the Syntetos-Boylan classification [75], making it suitable for a comprehensive robustness analysis.

The dataset spans 1,941 consecutive days, from 29 January 2011 to 19 June 2016, and covers 10 stores distributed across three US states: four stores in California, three in Texas, and three in Wisconsin. The product catalogue comprises 3,049 distinct items organised into three product categories: Foods (47.1%), Household (34.3%), and Hobbies (18.5%). At store-item level, the dataset contains 30,490 time series, each representing the daily unit sales of a single product in a single store. This is the level at which all models in this study are trained and evaluated.

The temporal evolution of these sales is captured in Figure 5.1, which shows aggregated daily sales by state over the full period. All three states exhibit a broadly stable demand regime with

moderate growth over time, with California consistently leading in sales volume given its larger store count.

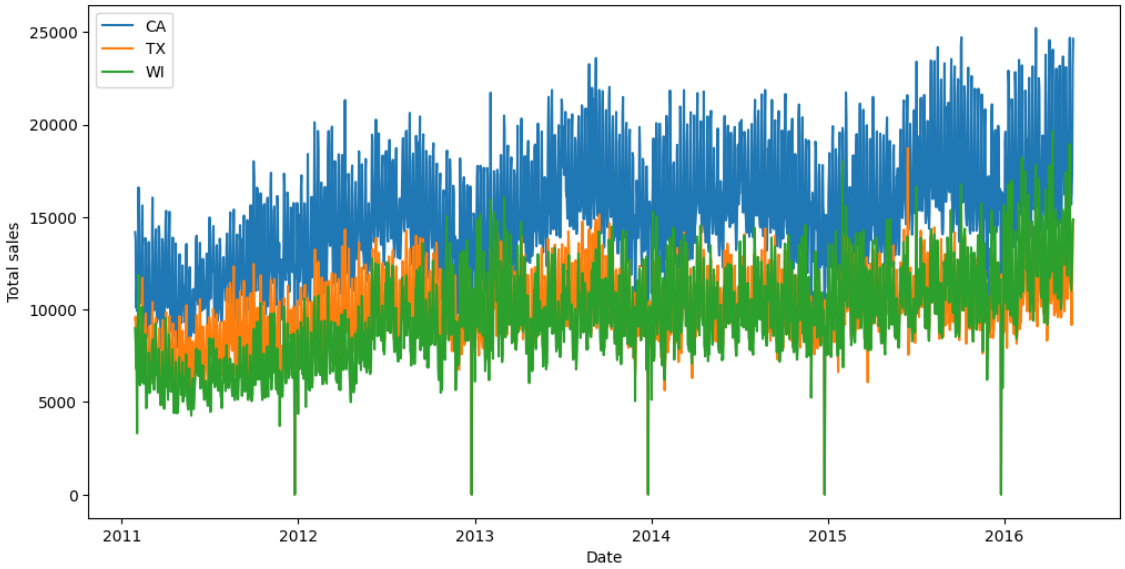


Figure 5.1: Aggregated daily unit sales by state over the full M5 dataset period.

The weekly seasonal profile, shown in Figure 5.2, confirms a consistent pattern across states and product categories: sales are lowest on Tuesday and Wednesday and peak on Saturday and Sunday, with the weekend uplift reaching approximately 20 to 25 percent above the weekly mean.

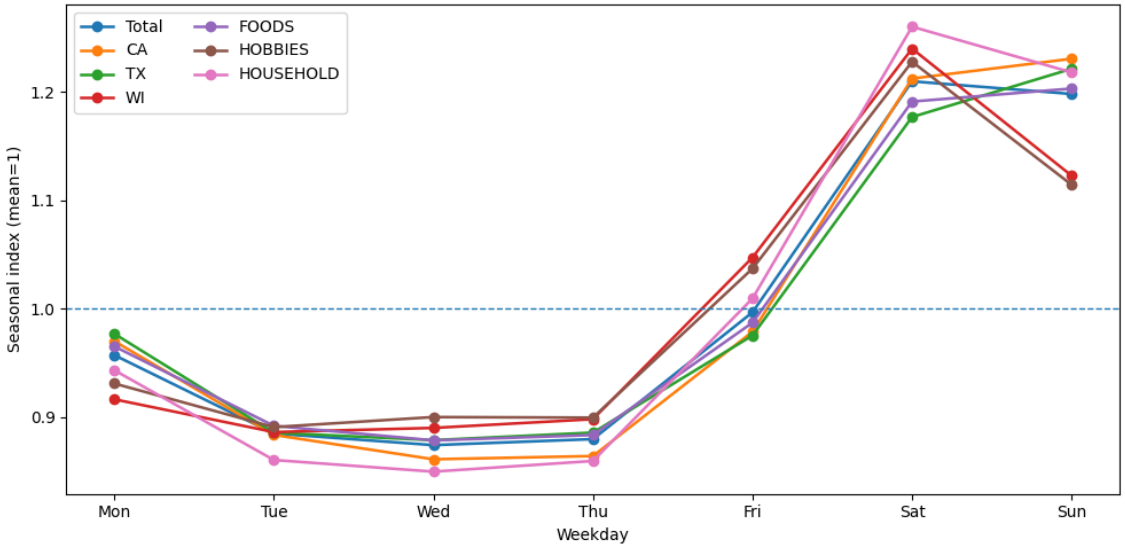
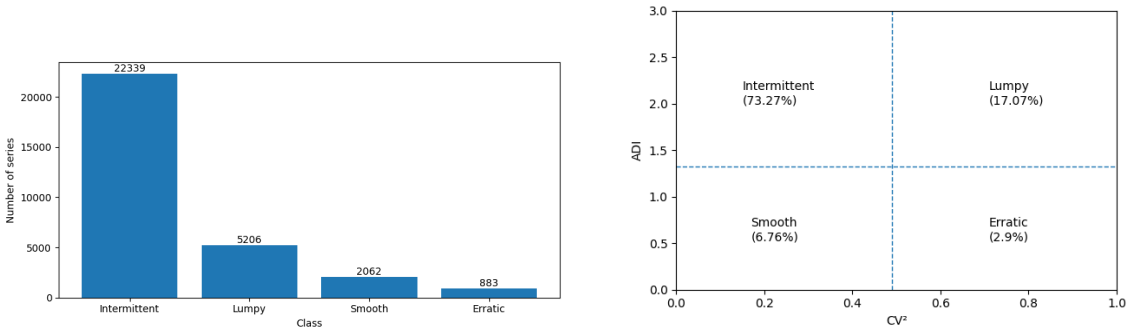


Figure 5.2: Weekly seasonal profile across states and product categories, expressed as a normalised index relative to the weekly mean.

The portfolio is dominated by intermittent demand patterns. Each series is classified according to the Syntetos-Boylan framework, which partitions demand into four classes based on two

dimensions: the Average Demand Interval (ADI), which measures how frequently demand occurs, and the Squared Coefficient of Variation of Non-zero Demand Sizes (CV^2), which captures size variability. Series with ADI above 1.32 and CV^2 below 0.49 are classified as Intermittent; those exceeding both thresholds are Lumpy; those below both thresholds are Smooth; and those with ADI below 1.32 but CV^2 above 0.49 are Erratic. Applying this classification to each of the 30,490 series reveals that 73.3% are Intermittent, 17.1% are Lumpy, 6.8% are Smooth, and only 2.9% are Erratic, as shown in Figures 5.3a and 5.3b.

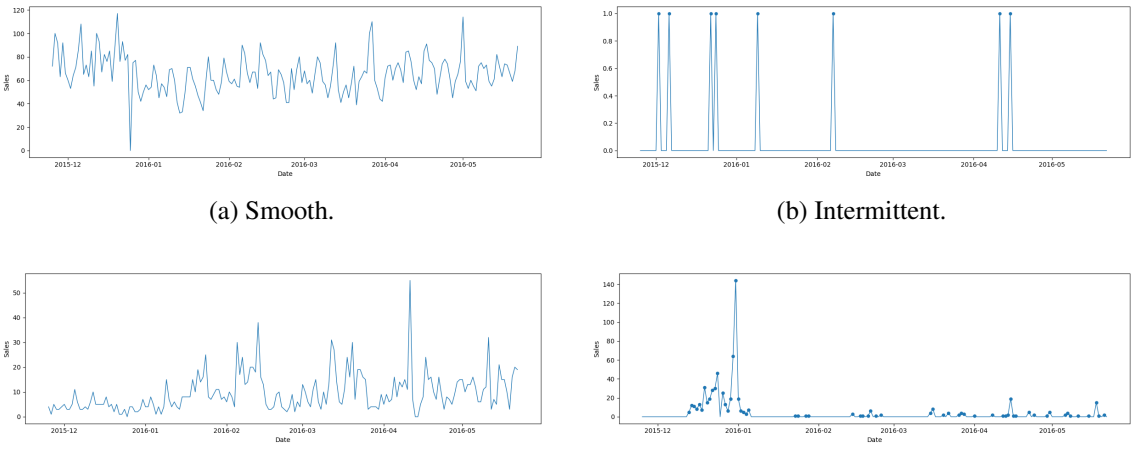


(a) Number of series per class.

(b) Distribution in the ADI-CV^2 space.

Figure 5.3: Demand classification of the 30,490 store-item series according to the Syntetos-Boylan framework.

Figure 5.4 illustrates a representative series from each class, making concrete what these classifications mean in practice: Smooth series exhibit dense, relatively stable sales; Intermittent series are characterised by frequent zero observations interspersed with consistent positive counts; Erratic series show dense but highly variable sales; and Lumpy series combine intermittency with large, irregular spikes. This distributional reality, where the vast majority of series are sparse and irregular, directly shapes the modelling choices adopted in this study.



(a) Smooth.

(b) Intermittent.

(c) Erratic.

(d) Lumpy.

Figure 5.4: Representative daily sales series for each demand class.

The M5 dataset also includes selling prices, promotional flags, and calendar variables such as holidays and SNAP purchase days. These variables are not used in the current study, as the evaluation focuses on the core forecasting task using sales history and categorical identifiers.

5.2 Feature Engineering

The input representation used across all three models is built from a small set of features. The goal was to capture recent demand dynamics whilst keeping the input space compact and consistent across architectures.

The temporal input comprises three variables: raw unit sales, a 7-day moving average, and a 28-day moving average. The raw sales signal carries the full granularity of daily demand, including the spikes, zeros, and noise that characterise store-item series at this level of aggregation. The moving averages serve a complementary purpose: the 7-day average captures short-term momentum, smoothing day-to-day fluctuations whilst remaining responsive to recent shifts in demand level, and the 28-day average provides a slower-moving reference that reflects the broader trend over the past four weeks. Both are computed on raw sales with a one-day lag, so that the value at day t uses only observations up to day $t-1$, avoiding any form of data leakage into the input features.

Two preprocessing decisions affect the length of usable history for each series. First, as a consequence of the one-day lag, the first 28 days of each series produce undefined values for the 28-day moving average and are discarded. Second, leading zeros preceding the first observed sale are removed, as they reflect the period before a product was introduced rather than genuine zero-demand days.

No additional scaling or normalisation is applied to the temporal features. All three variables are expressed in the same units as the target, namely daily unit sales, and their magnitudes are naturally comparable. Applying an external scaling transformation would alter the distributional properties of the inputs in a way that is inconsistent with the Tweedie loss, which is designed to operate on non-negative counts in their original scale. The internal scale handling performed by DeepAR-Direct through the factor v_i, described in Section 4.3.3, operates within the model. Inputs are scaled before entering the encoder and predictions are rescaled to the original units before the loss is computed, so the Tweedie loss always operates on values in the original scale.

The categorical input comprises two identifiers: item and store. These are not used as raw integers but are mapped to learned dense embeddings within each model. They allow each model to condition its temporal reasoning on the identity of the series being forecast, capturing latent structure across products and locations without requiring explicit hand-crafted features.

Table 5.1 summarises the full feature set used across all models.

Table 5.1: Input features used across all three models.

Feature	Type	Description	Availability
Sales	Temporal	Raw unit sales	Past only
MA-7	Temporal	7-day moving average	Past only
MA-28	Temporal	28-day moving average	Past only
Item ID	Categorical	Item identifier	Static
Store ID	Categorical	Store identifier	Static

5.3 Experimental Pipeline

Training each model involves two distinct phases: hyperparameter optimisation, conducted through a structured search over candidate configurations, and final training, in which the selected configuration is used to fit the model on all available data before generating forecasts. The following sections describe how data is partitioned to support this pipeline and how the optimisation is carried out.

5.3.1 Data Partitioning

The dataset is partitioned into three non-overlapping temporal segments: training, validation, and forecast. The forecast period corresponds to the 28 days immediately following the last observed day in the dataset, for which the final models generate predictions during inference. All 1,941 available days are used for training the final models, ensuring that the full historical signal is exploited before generating forecasts.

Hyperparameter optimisation requires an internal evaluation protocol that does not touch the forecast period. For this purpose, a temporal cross-validation scheme with four expanding folds is employed. A single train-validation split would evaluate the model on a single time window, which may not be representative given that demand patterns vary with seasonality and volatility across different periods of the year. By evaluating across four distinct validation windows, the cross-validation scheme produces a more robust estimate of generalisation performance and reduces the risk of selecting hyperparameters that overfit a particular temporal regime. This strategy follows the approach adopted by top-performing submissions in the M5 competition, where the use of at least four consecutive 28-day validation windows was identified as a reliable approximation of post-sample accuracy [51].

Each fold consists of a training segment that expands from day 1 to a fixed cutoff, followed by a contiguous 28-day validation block. The four folds are defined as follows: Fold 1 trains on days 1-1829 and validates on days 1830-1857; Fold 2 trains on days 1-1857 and validates on days 1858-1885; Fold 3 trains on days 1-1885 and validates on days 1886-1913; and Fold 4 trains on days 1-1913 and validates on days 1914-1941. This structure is illustrated in Figure 5.5.

Within each fold, training samples are generated through a sliding window procedure applied to every store-item series. Each sample consists of a 28-day lookback window followed by a 28-day prediction horizon. To reduce redundancy between consecutive samples whilst retaining

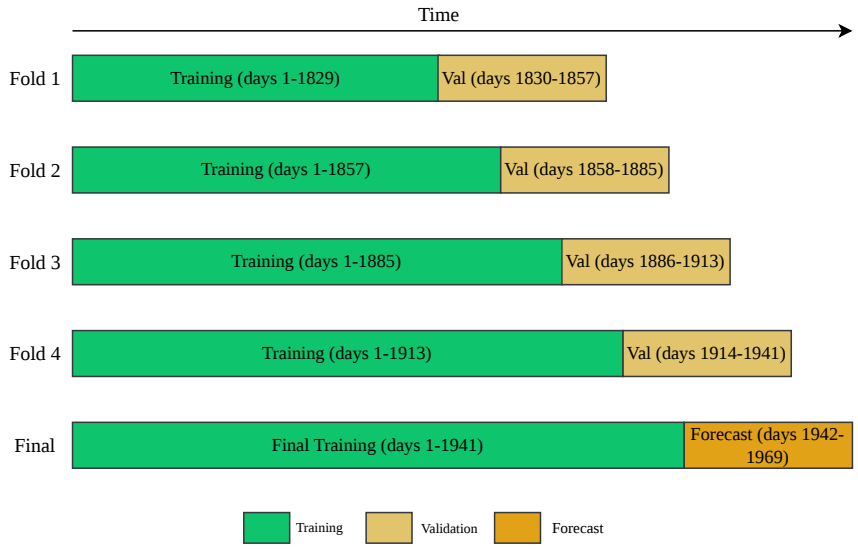


Figure 5.5: Expanding window cross-validation scheme used for hyperparameter optimisation. Each fold extends the training period by 28 days whilst maintaining a fixed 28-day validation block. The final model is trained on all available data before generating forecasts for the subsequent 28-day period.

sufficient coverage of the training period, a stride of 8 days is used: the window advances 8 days at a time rather than one, so that Window A covers days 1-28 to predict days 29-56, Window B covers days 9-36 to predict days 37-64, and so on. A stride of 7 was avoided, as it would cause every training window to begin on the same day of the week, introducing a systematic bias in the representation of weekly seasonality across samples. The stride of 8 breaks this alignment, ensuring that all days of the week are represented more uniformly throughout the training set. For validation, a single evaluation sample is generated per series, using the 28 days immediately preceding the validation block as the lookback window and the 28-day validation block itself as the prediction horizon.

5.3.2 Hyperparameter Optimisation

Hyperparameter optimisation is conducted using Optuna[1], a framework based on the Tree-structured Parzen Estimator (TPE) algorithm, which models the relationship between hyperparameter configurations and validation performance to guide the search towards promising regions of the search space. This is in contrast to grid or random search, which treat all configurations as equally likely candidates regardless of prior results. A total of 30 trials are run per model, with a fixed random seed of 42 to ensure reproducibility across runs.

Each trial is evaluated using the four-fold expanding window cross-validation scheme. Within each trial, the model is trained independently on each fold using the Adam optimiser, with early stopping applied to prevent overfitting: training is halted when the validation loss fails to improve for 10 consecutive epochs, up to a maximum of 150 epochs. Additionally, Optuna's median pruner

[1] https://optuna.org/, accessed 21 May 2026.

is employed to abort unpromising trials early: if a trial's validation loss at any fold exceeds the median of all completed trials at the same stage, the trial is terminated before completing the remaining folds, reducing the computational cost of the optimisation without affecting the quality of the final selection. The metric used to guide the optimisation is the mean Tweedie deviance loss computed on the 28-day validation block of each fold and averaged across the four folds. This is consistent with the training objective described in Section 4.2, ensuring that the hyperparameter selection criterion is aligned with the loss that the model directly minimises during training.

Once the best trial is identified, the number of training epochs for the final model is determined by taking the median of the best epoch recorded across the four folds of the winning trial. The final model is then retrained from scratch on all available data using the selected hyperparameters and this fixed epoch count, without any further validation.

Table 5.2 summarises the hyperparameter search space used across all models. The number of attention heads in the TFT is fixed at 4 across all experiments, and the Tweedie power parameter is fixed at $\rho = 1.3$ for all models; both are therefore excluded from the optimisation.

Table 5.2: Hyperparameter search space used during optimisation with Optuna.

Hyperparameter	**Type**	**Search space**	**Models**
Hidden size	Categorical	$\{64, 128, 256\}$	All
Batch size	Categorical	$\{64, 128, 256, 512\}$	All
Number of layers	Categorical	$\{1, 2, 3\}$	All
Dropout	Continuous	$[0.1, 0.5]$	All
Learning rate	Continuous	$[10^{-4}, 10^{-3}]$	All

5.4 Evaluation Metrics

Three metrics are used to evaluate forecast quality across all models: the Weighted Root Mean Squared Scaled Error (WRMSSE), the Mean Absolute Scaled Error (MASE), and Bias. Together they capture complementary aspects of model behaviour: weighted accuracy at portfolio level, scale-invariant performance relative to a seasonal baseline, and directional tendency of errors.

The WRMSSE is the official evaluation metric of the M5 competition and serves as the primary accuracy measure in this study. It is built upon the Root Mean Squared Scaled Error (RMSSE), which measures the forecast error of a single series relative to the error of a naïve one-step-ahead forecast computed over the training period. The RMSSE is defined in Eq. 5.1:

$$\text{RMSSE}_i = \sqrt{\frac{\frac{1}{h}\sum_{t=n+1}^{n+h}(y_t - \hat{y}_t)^2}{\frac{1}{n-1}\sum_{t=2}^{n}(y_t - y_{t-1})^2}} \tag{5.1}$$

where h is the forecast horizon, n is the length of the training period, y_t and $\hat{y}_t$ are the observed and predicted values respectively, and the denominator scales the error by the MSE of the naïve forecast over the training period, making the metric invariant to the magnitude of individual series.

The WRMSSE extends this by aggregating errors across all store-item series using a weighting scheme based on the relative sales volume of each series over the last 28 days of the training period, as defined in Eq. 5.2:

$$\text{WRMSSE} = \sum_{i=1}^{N} w_i \cdot \text{RMSSE}_i \tag{5.2}$$

where w_i is the normalised weight of series i. Series that contribute more to total revenue are therefore penalised more heavily for forecast errors, which reflects the operational reality that inaccurate forecasts for high-volume products have a greater economic impact than equivalent errors on slow-moving items.

The MASE is computed using a seasonal naïve benchmark with a lag of 7 days, reflecting the weekly periodicity identified in Section 5.1. Unlike scale-dependent metrics such as MAE, the MASE is invariant to the magnitude of individual series, making it suitable for comparing performance across the heterogeneous store-item portfolio of the M5 dataset. It is defined in Eq. 5.3:

$$\text{MASE} = \frac{\frac{1}{h}\sum_{t=1}^{h} |y_t - \hat{y}_t|}{\frac{1}{n-m}\sum_{t=m+1}^{n} |y_t - y_{t-m}|} \tag{5.3}$$

where $m = 7$ is the seasonal period, h is the forecast horizon, and the denominator is the mean absolute error of the seasonal naïve forecast over the training period of length n.

Bias measures the tendency of a model to systematically overestimate or underestimate demand, and is defined as the mean signed error across all series and horizon steps, as shown in Eq. 5.4:

$$\text{Bias} = \frac{1}{N \cdot h} \sum_{i=1}^{N} \sum_{t=1}^{h} (\hat{y}_{i,t} - y_{i,t}) \tag{5.4}$$

Unlike the WRMSSE and MASE, Bias is expressed in the original units of the target variable and is therefore scale-dependent. Its purpose is not to compare performance across individual series but to characterise the directional tendency of errors at portfolio level. A positive Bias indicates that the model consistently overpredicts demand, whilst a negative value indicates underprediction. In retail forecasting, systematic bias is operationally costly regardless of its direction: overprediction leads to excess inventory and waste, whilst underprediction results in stockouts and lost sales. Reporting Bias alongside accuracy metrics therefore provides a more complete characterisation of model behaviour than error magnitude alone.

5.5 Computational Infrastructure

All experiments were conducted on a workstation provided by the Artificial Intelligence and Computer Science Laboratory of the University of Porto. The machine is equipped with two Intel Xeon 6258R processors running at 2.7 GHz, 128 GB of RAM, and a single NVIDIA Quadro RTX 8000 GPU with 48 GB of VRAM. All model training and inference was performed on this GPU. The software environment was built on PyTorch[2] 2.5.1 with CUDA 12.1. Automatic Mixed Precision

[2] https://pytorch.org/, accessed 22 May 2026.

training was enabled throughout, reducing memory consumption and accelerating computation by performing forward and backward passes in 16-bit floating point whilst maintaining 32-bit precision for parameter updates.

Chapter 6

Experiments and Results

This chapter presents the empirical results of the three research questions investigated in this dissertation. Section 6.1 addresses the first research question by evaluating the predictive accuracy of the three DL models against a set of classical statistical baselines, global ML methods, and two top-performing solutions from the M5 Accuracy competition. Section 6.2 addresses the second research question by examining how the computational cost and forecasting accuracy of each model evolve as the number of stores in the training corpus increases. Section 6.3 addresses the third research question by decomposing model performance across the four demand pattern classes defined by the Syntetos-Boylan framework and examining the time series properties most strongly associated with forecast difficulty within each class.

6.1 Accuracy Analysis

This section addresses the **first research question** by evaluating the predictive accuracy of the three DL models evaluated in this dissertation against a set of classical statistical baselines, global ML methods, and two top-performing solutions from the M5 Accuracy competition. The evaluation uses the full ten-store dataset, with performance assessed through the WRMSSE, MASE, and Bias metrics.

6.1.1 Baseline Models

Five reference methods are used to establish a performance floor against which the proposed DL models are evaluated. These methods were selected from the set of benchmarks provided by the M5 competition organisers, whose descriptions and implementation code are publicly available in the official competition repository[1]. The selection covers the main families of classical forecasting approaches: a naïve seasonal method, a statistical model with automatic component selection, an autoregressive integrated model, and two global ML methods. All baselines are evaluated at the store-product level.

[1] https://drive.google.com/drive/folders/1D6EWdVSaOtrP1LEFh1REjI3vej6iUS_4, accessed 26 May 2026.

6.1.1.1 Seasonal Naïve

The Seasonal Naïve baseline produces forecasts by repeating the last observed weekly cycle across the forecast horizon. For a horizon of 28 days, the final seven observed values are tiled four times, so that each future day inherits the sales count recorded on the corresponding day of the most recent week. The method is applied independently to each store-item series and requires no parameter estimation, capturing only the most immediate weekly pattern without any smoothing or trend component.

6.1.1.2 Exponential Smoothing

The Exponential Smoothing (ES) baseline applies the `es()` function from the `smooth` R package, which automatically selects the most appropriate state space model from the Error, Trend, Seasonality (ETS) family using information criteria. A weekly seasonality period of 7 is specified, allowing the selected model to capture day-of-week demand patterns. The method is applied independently to each store-item series and requires no manual specification of model components. Model parameters are estimated by maximum likelihood.

6.1.1.3 Autoregressive Integrated Moving Average

The ARIMA baseline applies the `auto.arima()` function from the `forecast` R package, which automatically selects the order of the autoregressive, differencing, and moving average components using information criteria. A weekly seasonality period of 7 is specified, allowing the function to consider both non-seasonal ARIMA and SARIMA configurations. The selection between these is made independently for each store-item series, meaning the resulting model family varies across the portfolio depending on the seasonal structure identified in each series. Model parameters are estimated by maximum likelihood, with order selection based on the Akaike Information Criterion with correction for finite sample sizes.

6.1.1.4 Global Multi-Layer Perceptron

The Global Multi-Layer Perceptron (GMLP) baseline trains a single MLP model jointly across all store-item series, enabling cross-series learning rather than fitting independent models per series. Each training sample consists of the last 14 observed values as input features, supplemented by two demand characterisation statistics: the ADI and the CV^2. These statistics provide the model with information about the intermittency and size variability of each series, facilitating learning across series with heterogeneous demand patterns. Input values are normalised to the $[0, 1]$ range using per-series min-max scaling prior to training, and predictions are rescaled to the original units at inference time. The network has a single hidden layer with 28 neurons and a logistic activation function, trained by minimising the MSE using the Scaled Conjugate Gradient algorithm. To reduce sensitivity to random weight initialisation, 10 independent runs are performed and the median forecast across runs is used as the final prediction.

6.1.1.5 Global Random Forest

The Global Random Forest (GRF) baseline follows the same global training procedure as the GMLP, using identical input features, normalisation, and sampling strategy. The MLP is replaced by a RF of 500 decision trees, with node splitting performed by minimising the MSE. The final forecast is taken as the median across the predictions of the ensemble.

6.1.2 M5 Competition Participants

To contextualise the results of the proposed models within the broader landscape of the M5 Accuracy competition, two participating solutions are described in this subsection. These were selected to represent distinct methodological approaches: the first-placed solution employs gradient boosting with a structured pooling strategy, whilst the third-placed solution adopts a DL approach based on a modified autoregressive recurrent architecture. Both solutions are evaluated at the store-product level.

6.1.2.1 Direct and Recursive Forecast Averaging via Partial Pooling

The winning solution presented in [31] proposes the Direct and Recursive Forecast Averaging Method via Partial Pooling (DRFAM). The method constructs base forecasting models using Light Gradient Boosting Machine (LightGBM), a gradient boosting framework, trained across three levels of partial pools: store, store-category, and store-department. For each of the 10 stores, this yields 11 data pools (1 store pool, 3 store-category pools, and 7 store-department pools), totalling 110 pools across the full dataset. All base models are trained at the store-product level, regardless of the pool level used.

Two types of base models are constructed for each pool. Direct forecasting models predict all 28 horizon steps using only known historical observations, with the most recent available sales starting from $t - 28$. Recursive forecasting models condition on previously forecasted values. Rolling statistics are computed over shorter recent windows and fed back as input features for subsequent steps. The final DRFAM submission averages the forecasts of all six base models per product (three direct and three recursive, one from each pool level), obtaining complementary effects between the two forecasting strategies.

Features used across all base models fall into six categories: product and store identifiers, calendar variables, event indicators, price-derived statistics, lagged sales statistics over multiple window lengths and recursive sales statistics. The Tweedie loss function is used as the training objective, motivated by the high proportion of zero sales values in the dataset. Model selection is performed through time series cross-validation over 13 consecutive 28-day validation windows.

6.1.2.2 Robust Recurrent Network for Intermittent Forecasting

The third-placed solution presented in [33] applies the Rolled DeepAR architecture described in Section 3.2.3 to the M5 dataset. During the rolling training procedure, samples drawn from the

predicted Tweedie distribution are used as inputs, which simultaneously corrects for exposure bias and increases the diversity of training sequences. The network consists of two stacked LSTM layers with 120 hidden units each, trained using the negative log-likelihood of the Tweedie distribution as the loss function. The input window spans 28 days and includes 100 feature channels covering sales history, moving averages, time features, price information, SNAP indicators, calendar events, item and store identifiers, and a zero-sale period counter indicating the number of consecutive days without sales. The model is trained globally across all store-item series using the Adam optimiser with a learning rate schedule that increases linearly for the first five epochs and decays via cosine annealing thereafter.

Model selection is performed by evaluating each candidate model over 14 consecutive 28-day validation windows, using the average WRMSSE across all hierarchical levels as the selection criterion. The final submission is an ensemble of 43 models combining two configurations that differ in Tweedie power parameter and dropout settings, with predictions averaged uniformly across all ensemble members.

6.1.3 Model Configurations

The hyperparameter configurations selected for each architecture reflect the best solution found for that specific model, and the differences between them reveal how each architecture allocates its representational capacity. The DeepAR-Direct and LSTM both select a hidden size of 64 and converge to deeper configurations with multiple stacked layers, suggesting that these architectures benefit more from additional recurrent depth than from wider hidden representations. The TFT, by contrast, selects a wider representation of 128 with a single recurrent layer, suggesting that its attention mechanism and gating operations benefit from a larger hidden dimension to distribute information effectively across its multiple processing pathways. The compact hidden size of the DeepAR-Direct reflects the efficiency of its encoder-decoder design: rather than compressing the entire input sequence into a single fixed-length vector, the decoder processes the forecast horizon step by step, reducing the representational burden placed on any individual hidden state. The dropout rates vary considerably across models, with the DeepAR-Direct selecting the highest value. Table 6.1 summarises the selected hyperparameters for each model.

Table 6.1: Hyperparameter configurations selected by Optuna for the LSTM, TFT, and DeepAR-Direct models.

Hyperparameter	**LSTM**	**TFT**	**DeepAR-Direct**
Hidden Size	64	128	64
Heads	–	4	–
Layers	3	1	3
Batch Size	64	128	256
Dropout	0.2517	0.1727	0.4878
LR ($\times 10^{-4}$)	7.86	1.53	8.90

6.1.4 Results

Table 6.2 summarises the forecasting performance of all models evaluated at the store-product level, ranked by WRMSSE. Figure 6.1 presents the percentage improvement in WRMSSE and MASE over the Seasonal Naïve baseline for all models, providing a visual comparison of relative gains.

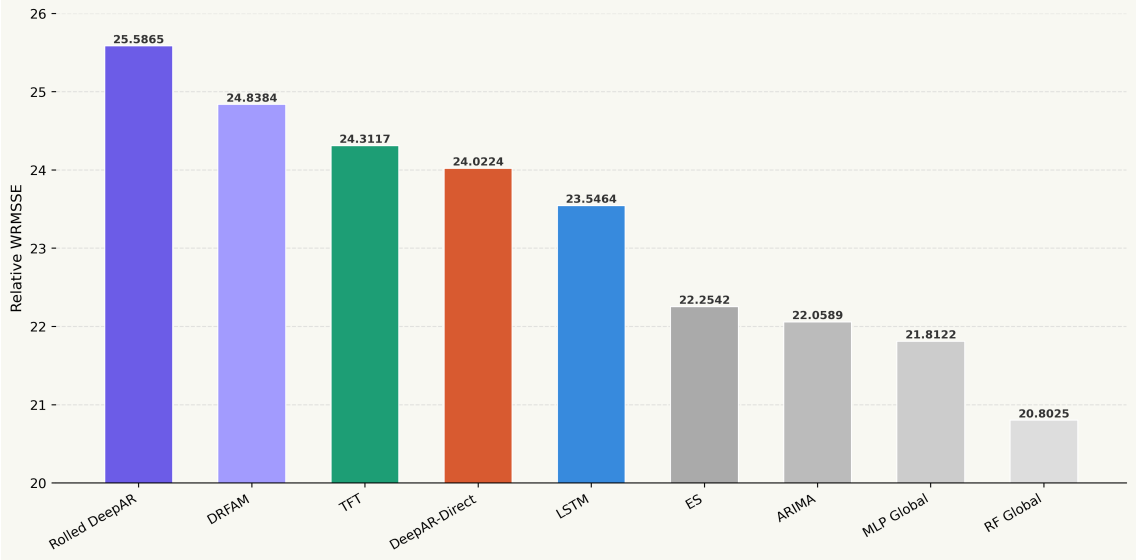
Table 6.2: Comparison of forecasting models performance based on WRMSSE, MASE, and Bias.

Model	**WRMSSE**	**MASE**	**Bias**
Rolled DeepAR	0.8754	1.0569	0.0039
DRFAM	0.8842	1.0518	0.0026
TFT	0.8905	1.0577	-0.0467
DeepAR-Direct	0.8939	1.0805	0.0108
LSTM	0.8995	1.0749	-0.0100
ES	0.9147	1.0623	-0.0046
ARIMA	0.9170	1.0667	0.0083
MLP Global	0.9199	1.1265	0.0092
RF Global	0.9318	1.1117	0.1157
Seasonal Naïve	1.1764	1.2236	-0.0172

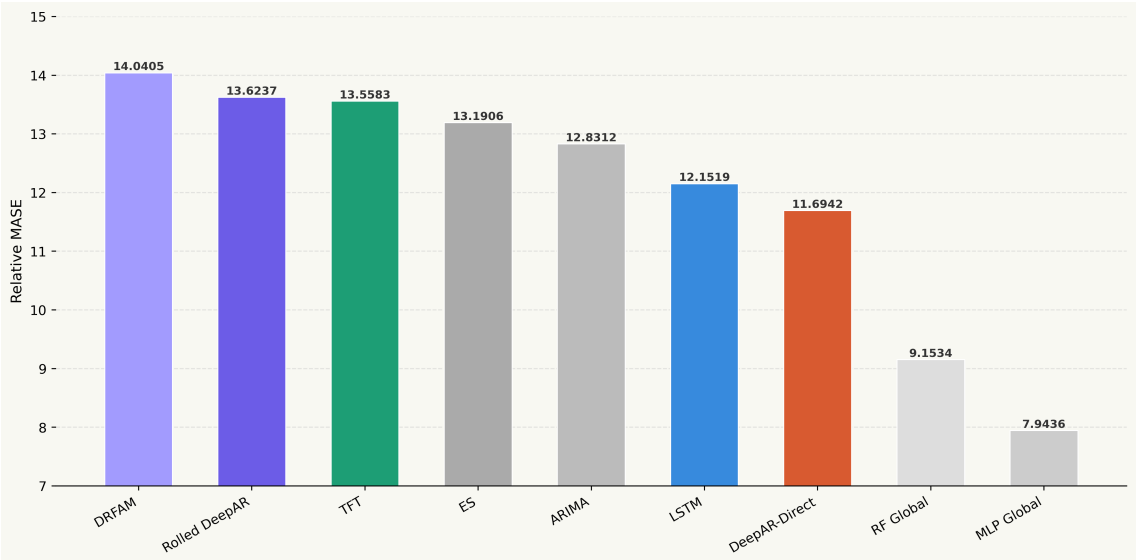
All three DL models proposed in this dissertation outperform the classical statistical baselines and the global ML methods on WRMSSE, with TFT achieving the lowest value among them at 0.8905, followed by DeepAR-Direct at 0.8939 and LSTM at 0.8995. The two M5 competition participants rank above all models evaluated in this study, with Rolled DeepAR achieving 0.8754 and DRFAM 0.8842, representing improvements of 25.59% and 24.84% over the Seasonal Naïve baseline respectively, as illustrated in Figure 6.1a. Among the classical baselines, ES and ARIMA perform comparably at 0.9147 and 0.9170, whilst the global ML methods trail slightly behind, with GMLP at 0.9199 and GRF at 0.9318. The Seasonal Naïve baseline records the highest WRMSSE at 1.1764, serving as the lower bound of performance in this comparison.

The MASE rankings partially diverge from those of the WRMSSE, as illustrated in Figure 6.1b. Whilst TFT retains its position as the best-performing model among those proposed in this dissertation at 1.0577, DRFAM achieves the lowest MASE overall at 1.0518, ranking above Rolled DeepAR on this metric despite placing below it on WRMSSE. Among the classical baselines, ES records the lowest MASE at 1.0623, outperforming ARIMA at 1.0667. DeepAR-Direct and LSTM rank below ES and ARIMA on MASE despite outperforming them on WRMSSE, with values of 1.0805 and 1.0749 respectively. The global ML methods record the highest MASE values among non-naïve models, with GMLP at 1.1265 and GRF at 1.1117.

Regarding directional bias, the models differ in their tendency to overpredict or underpredict demand. TFT and LSTM exhibit negative bias values of -0.0467 and -0.0100 respectively, indicating a systematic tendency to underpredict. DeepAR-Direct, ARIMA, and GMLP show small positive bias values, reflecting a slight overprediction tendency. GRF records the largest positive bias at 0.1157, suggesting a more pronounced tendency to overestimate demand across



(a) Percentage improvement in WRMSSE over the Seasonal Naïve baseline, ranked in descending order.



(b) Percentage improvement in MASE over the Seasonal Naïve baseline, ranked in descending order.

Figure 6.1: Percentage improvement in WRMSSE and MASE over the Seasonal Naïve baseline for all evaluated models. The Seasonal Naïve baseline is excluded from both charts as it serves as the reference point.

the portfolio. The two M5 competition participants display near-zero bias values of 0.0039 and 0.0026, indicating well-calibrated directional behaviour.

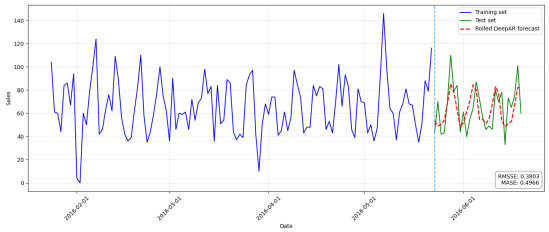
Figure 6.2 illustrates the forecasts generated by all models for a representative high-volume series, item FOODS_3_090 from store TX_3. This series exhibits a Smooth demand pattern with pronounced weekly seasonality and occasional demand spikes. The DL models and the two M5 participants track the weekly oscillations of the test period more closely than the classical baselines, with Rolled DeepAR and DRFAM producing the most accurate trajectories as reflected by their RMSSE values of 0.3803 and 0.4210 respectively. Among the models proposed in this dissertation, DeepAR-Direct achieves the lowest RMSSE on this series at 0.3851, slightly outperforming TFT at 0.3989 and LSTM at 0.4041. The classical statistical models fail to capture the magnitude and timing of demand fluctuations: ARIMA produces a smooth forecast that underestimates peak values, whilst ES overshoots in the latter part of the horizon. The global ML methods generate near-constant forecasts that fail to reproduce the weekly pattern, with GRF and GMLP recording RMSSE values of 0.5035 and 0.4773 respectively. The Seasonal Naïve forecast replicates the weekly cycle mechanically but cannot adapt to the level shift visible in the test period, resulting in the highest RMSSE of 0.6215 on this series.

6.1.5 Discussion

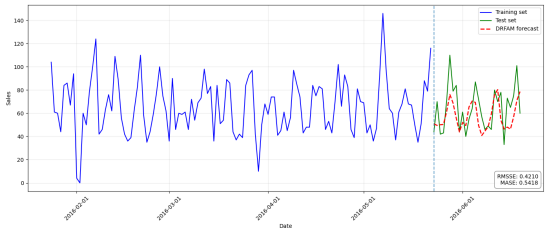
All three DL models proposed in this study outperform every classical statistical baseline and global ML method on WRMSSE, confirming that global neural architectures trained jointly across thousands of store-item series can extract predictive signal that series-by-series approaches cannot. The margin over ES and ARIMA, whilst modest in absolute terms (approximately 0.02 to 0.09 WRMSSE points), is consistent across all three DL models and reflects a structural advantage. By sharing parameters across series, the DL models leverage demand patterns common to multiple products and stores, a form of cross-series learning that local statistical models are unable to exploit. The Tweedie training objective reinforces this advantage by aligning the loss function with the distributional properties of retail demand, penalising errors in proportion to the skewness and sparsity of each series rather than treating all deviations symmetrically as MSE-based objectives would.

The global ML methods, despite also training a single model across all series, underperform the DL models on both WRMSSE and MASE. This suggests that the representational capacity of the LSTM encoder and the structured input processing of the TFT provide a meaningful advantage over the shallow architectures of the GMLP and GRF, particularly in capturing the temporal dynamics of demand over the 28-day horizon. The near-constant forecasts produced by GRF and GMLP for the representative series in Figure 6.2 illustrate this limitation concretely: without a sequential inductive bias, these models reduce to predicting a smoothed level rather than tracking the weekly oscillations visible in the data.

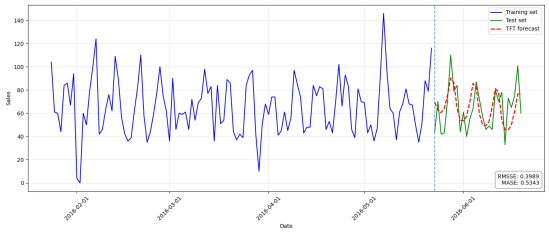
The divergence between WRMSSE and MASE rankings warrants careful interpretation. DeepAR-Direct and LSTM rank below ES and ARIMA on MASE despite outperforming them on WRMSSE. Since WRMSSE weights each series by its dollar sales volume whilst MASE treats all series



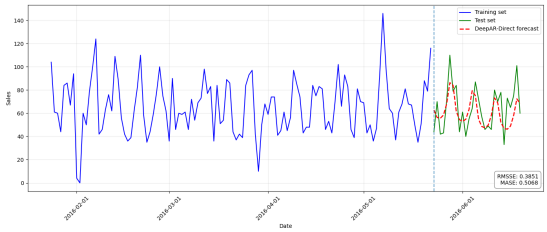
(a) Rolled DeepAR



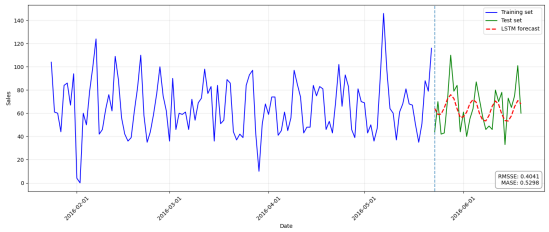
(b) DRFAM



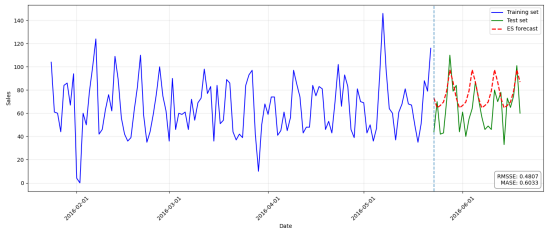
(c) TFT



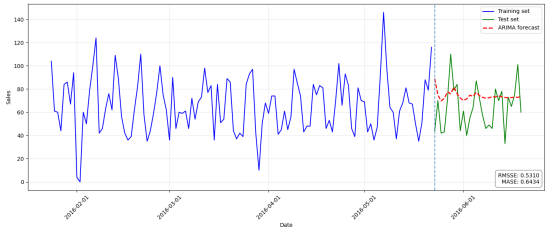
(d) DeepAR-Direct



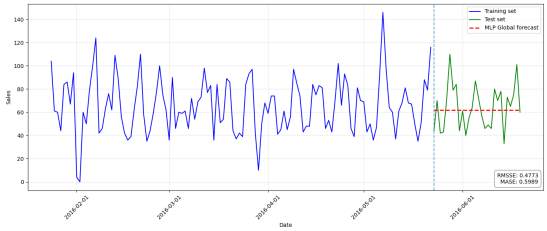
(e) LSTM



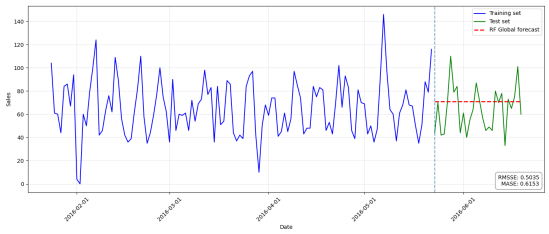
(f) Exponential Smoothing



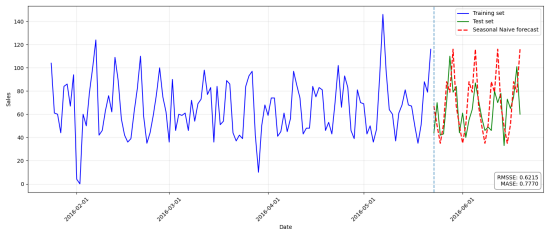
(g) ARIMA



(h) MLP Global



(i) Random Forest Global



(j) Seasonal Naïve

Figure 6.2: Forecast trajectories of all evaluated models for item FOODS_3_090 from store TX_3, a high-volume Smooth series. The blue line represents the training period, the green line the test period, and the red dashed line the model forecast. RMSSE and MASE values for each model are shown in the bottom-right corner of each panel.

equally, this pattern suggests that the gains of the DL models are concentrated in higher-volume series. For low-volume and intermittent series, which constitute 73.3% of the M5 portfolio, the advantage of the DL models over classical baselines is smaller or absent, as the sparse and irregular demand signal provides limited information for the recurrent encoder to exploit.

The observation that all models record MASE values above 1.0 does not imply that any of them underperforms the Seasonal Naïve in the test period. The MASE denominator is computed over the training period, and its scale therefore reflects the average in-sample one-step naïve error rather than the out-of-sample difficulty of the test window. The Seasonal Naïve itself records a MASE of 1.2236 in the test period, which confirms that all evaluated models outperform it on this metric as well. The elevated MASE values across the board reflect the intrinsic difficulty of forecasting at store-item granularity in a portfolio dominated by intermittent demand, not a failure of the models relative to the naïve benchmark.

The directional bias patterns reveal meaningful differences in how each model responds to the distributional properties of retail demand. The underprediction tendency of TFT may reflect the interaction between its attention mechanism and the Tweedie objective: by distributing weight across a wider temporal context, the multi-head attention layer tends to smooth out demand peaks, systematically underestimating high-sales days. The pronounced overprediction of GRF is a direct consequence of its inability to track demand oscillations. The near-zero bias of the two M5 competition participants suggests that their richer feature sets, particularly price and promotional information, help the models anticipate demand shifts in both directions more accurately.

The gap between the models proposed in this dissertation and the two M5 competition participants is small given the substantial differences in experimental setup. Rolled DeepAR and DRFAM both incorporate substantially richer feature sets, including price information, promotional flags, SNAP indicators, and calendar events, none of which are used in this study. Furthermore, both competition solutions employ large ensembles of independently trained models, a strategy that systematically reduces variance at the cost of computational overhead. The fact that the three models proposed here, trained on sales history and categorical identifiers alone, achieve WRMSSE values within 0.015 to 0.024 points of the competition winners suggests that the core architectural choices and training objective account for a large portion of the attainable accuracy. The marginal contribution of additional features and ensembling, whilst real, is more modest than the complexity of those solutions might suggest.

RQ1: The three DL models proposed in this dissertation consistently outperform all classical statistical baselines and global ML methods on WRMSSE at the store-product level, with improvements ranging from 1.7 to 3.0 percentage points over the best classical baseline. These gains are most pronounced for high-volume series, where the temporal representations learned by the DL models provide an advantage over series-by-series statistical approaches. The results confirm that global DL architectures trained on real operational data can improve demand forecasting accuracy in the food retail sector, although the magnitude of improvement is moderated by the dominance of intermittent demand patterns in the portfolio, which limit the informative signal available to the recurrent encoders.

6.2 Scalability Analysis

This section addresses the **second research question** by evaluating how the computational cost and forecasting accuracy of the three DL models developed in this dissertation evolve as the number of stores included in the training corpus increases. Performance is assessed through both the accuracy metrics and a set of computational metrics covering training time, inference time, memory consumption, parameter count, and Floating Point Operations (FLOPs) estimates.

6.2.1 Experimental Setup

To evaluate how each model scales with the size of the training corpus, four scalability scenarios are defined, each corresponding to a distinct subset of the M5 dataset of increasing size. The three architectures evaluated in this study, LSTM, TFT, and DeepAR-Direct, are trained and evaluated independently under each scenario, allowing their computational and predictive behaviour to be observed as the number of stores, and consequently the number of time series, grows.

The scenarios are constructed by varying the number of stores included in the training and evaluation corpus, from a single store in Scenario 1 to the full ten-store dataset in Scenario 4. Since each store contributes exactly $3,049$ store-item series to the dataset, the total number of series scales linearly with the number of stores, ranging from $3,049$ in Scenario 1 to $30,490$ in Scenario 4. Within each scenario, all series belonging to the selected stores are included without filtering, preserving the full heterogeneity of demand patterns present in each subset.

Store selection was performed randomly within each scenario, with the constraint that the stores selected for a given scenario are not necessarily a superset of those in the preceding one. This design choice reflects the goal of evaluating scalability as a function of dataset size rather than as a cumulative expansion of a fixed core, avoiding the confound that would arise if each scenario simply added stores to the previous one. The stores assigned to each scenario are listed in Table 6.3.

Across all four scenarios, the training and evaluation protocol remains identical to that described in Section 5.3: the same expanding window cross-validation scheme is used for hyperparameter optimisation, the same final training procedure is applied, and forecasts are generated for

Table 6.3: Store composition and series count for each scalability scenario.

Scenario	Stores	Series
1	CA_1	3,049
2	CA_4, TX_1, WI_1	9,147
3	CA_1, CA_4, TX_2, WI_2, WI_3	15,245
4	CA_1, CA_2, CA_3, CA_4, TX_1, TX_2, TX_3, WI_1, WI_2, WI_3	30,490

the same 28-day horizon. This consistency ensures that any differences in computational cost or predictive accuracy observed across scenarios are attributable to the change in dataset size rather than to differences in the experimental protocol. Hyperparameters are optimised independently for each model and each scenario, allowing each configuration to adapt to the scale of the data it is trained on rather than inheriting a fixed configuration from a different setting.

6.2.2 Scalability Metrics

Evaluating scalability requires looking beyond predictive accuracy alone. As the number of stores grows, so does the volume of data on which each model must be trained, the number of series over which inference must be performed, and the computational resources required to support both. To characterise these dimensions systematically, a set of metrics is defined that together capture the cost of training and deploying each model under each of the four scalability scenarios. These metrics fall into two broad categories: those that describe the cost of the training phase, and those that describe the cost of inference.

On the training side, five quantities are recorded for each model and scenario. The total training time, expressed in seconds, measures the wall-clock duration of the final training run from the first epoch to convergence, excluding hyperparameter optimisation. This reflects the computational effort required to fit the model on a given dataset and is the most direct measure of training cost. The number of epochs to convergence is also recorded, capturing how many full passes through the training data were required before the early stopping criterion was met. Together with the batch size, this quantity determines the total number of parameter updates performed during training and therefore contextualises the observed training times. Alongside this, the peak and average GPU VRAM consumption during training are recorded in megabytes, capturing respectively the maximum memory demand at any point during the run and the typical memory footprint sustained throughout. These two quantities are complementary: peak VRAM determines whether a model can be trained on a given GPU at all, whilst average VRAM reflects the efficiency with which memory is utilised over the course of training. Finally, the number of trainable parameters is counted for each model and scenario using PyTorch's standard parameter enumeration, summing the element counts of all tensors for which gradient computation is enabled. Since the item and store embedding layers grow with the number of unique identifiers in the dataset, the parameter count is not fixed across scenarios and is therefore treated as a scalability metric.

The computational cost of a single forward pass is estimated using the thop[2] library, which instruments the model's computational graph and accumulates floating-point operation counts across all layers. Because the cost of a single forward pass determines the computational burden of both training and inference, it is reported separately as a metric shared between both phases. For training, the estimated FLOPs are obtained by multiplying the forward pass cost by three [36], following the standard approximation that the backward pass incurs approximately twice the cost of the forward pass due to the computation of gradients with respect to both activations and parameters. For inference, the raw forward pass cost is reported directly. In both cases, the FLOPs value is reported for the full batch used during profiling.

Applying thop to the three architectures requires different levels of instrumentation. For the LSTM and DeepAR-Direct, the standard thop profiler is sufficient, as both architectures are composed entirely of operations that thop tracks natively. The TFT, however, contains several custom modules whose internal operations are not visible to the standard profiler: the interpretable multi-head attention mechanism, which performs explicit query-key and attention-value matrix multiplications outside the standard nn.MultiheadAttention[3] interface; the gated linear units, whose sigmoid activations and element-wise multiplications are not captured by the linear layer hooks; and the GateAddNorm modules, whose residual additions fall outside the default operation registry. For each of these, a dedicated FLOPs counter is registered with thop prior to profiling, ensuring that the resulting estimates are architecturally complete and comparable across models.

On the inference side, two additional quantities are recorded. The total inference time measures the wall-clock duration required to generate forecasts for all series in a given scenario in a single pass, expressed in seconds. This reflects the latency a deployed system would incur when producing a full set of store-item forecasts, and scales naturally with the number of series as the scenario grows. The peak GPU VRAM consumption during inference is also recorded, capturing the maximum memory demand encountered when processing the full series portfolio. Unlike training VRAM, inference VRAM is typically lower and grows more predictably with batch size, but remains a relevant constraint in resource-limited deployment environments.

Table 6.4 summarises all metrics used in this analysis, together with their phase of measurement and a brief description of what each captures.

6.2.3 Results

Table 6.5 presents the hyperparameter configurations selected by Optuna for each model and scenario. These configurations vary across scenarios, reflecting the fact that the optimal architectural choices depend on the scale of the training corpus. The hidden size of the TFT grows from 64 in Scenario 1 to 256 in Scenario 3 before returning to 128 in Scenario 4, whilst the LSTM selects a hidden size of 256 across the first three scenarios but drops to 64 in Scenario 4. The DeepAR-Direct consistently selects a hidden size of 64 across all scenarios, suggesting that its

[2] https://pypi.org/project/thop/, accessed 1 Jun 2026.

[3] https://docs.pytorch.org/docs/2.12/generated/torch.nn.MultiheadAttention.html, accessed 2 Jun 2026.

Table 6.4: Summary of scalability metrics used in the analysis.

Metric	**Phase**	**Description**
Total training time (s)	Training	Wall-clock duration of the final training run
Epochs to convergence	Training	Number of training epochs before early stopping
Peak VRAM (MB)	Training	Maximum GPU memory consumption during training
Average VRAM (MB)	Training	Mean GPU memory consumption during training
Trainable parameters	Training	Total number of learnable parameters
FLOPs training est.	Training	Forward pass FLOPs $\times$ 3, reported per batch
FLOPs inference (forward)	Inference	Forward pass FLOPs reported per batch
Total inference time (s)	Inference	Wall-clock duration to forecast all series
Peak VRAM (MB)	Inference	Maximum GPU memory consumption during inference

encoder-decoder structure achieves sufficient representational capacity without requiring larger hidden dimensions. The number of layers similarly varies: the LSTM selects two layers in Scenarios 1 through 3 but increases to three in Scenario 4, whilst the TFT consistently converges to a single-layer configuration and the DeepAR-Direct selects two layers in Scenario 1 and three in all subsequent scenarios. The batch size shows the most pronounced variation, ranging from 64 to 512 across models and scenarios. This variation has direct consequences for the interpretation of training time and FLOPs, and is an important contextual factor when comparing computational metrics across scenarios.

Table 6.5: Hyperparameter configurations selected by Optuna for each model and scenario.

Hyperparameter	**Scenario 1**			**Scenario 2**			**Scenario 3**			**Scenario 4**		
	LSTM	**TFT**	**DeepAR-Direct**	**LSTM**	**TFT**	**DeepAR-Direct**	**LSTM**	**TFT**	**DeepAR-Direct**	**LSTM**	**TFT**	**DeepAR-Direct**
Hidden Size	256	64	64	256	128	64	256	256	64	64	128	64
Heads	–	4	–	–	4	–	–	4	–	–	4	–
Layers	2	1	2	2	1	3	2	1	3	3	1	3
Batch Size	64	64	256	64	256	512	512	512	256	64	128	256
Dropout	0.4084	0.2066	0.2849	0.3525	0.3430	0.3496	0.1030	0.3254	0.4919	0.2517	0.1727	0.4878
LR ($\times 10^{-4}$)	3.12	3.02	9.78	9.81	1.48	9.55	2.79	2.42	9.27	7.86	1.53	8.90

Table 6.6 presents the forecasting accuracy of each model across the four scalability scenarios. All three models exhibit a consistent degradation in WRMSSE as the number of stores increases, rising from values in the range 0.843–0.853 in Scenario 1 to values in the range 0.890–0.900 in Scenario 4.

Table 6.6: Forecasting accuracy of each model across the four scalability scenarios.

Metric	**Scenario 1**			**Scenario 2**			**Scenario 3**			**Scenario 4**		
	LSTM	**TFT**	**DeepAR-Direct**	**LSTM**	**TFT**	**DeepAR-Direct**	**LSTM**	**TFT**	**DeepAR-Direct**	**LSTM**	**TFT**	**DeepAR-Direct**
WRMSSE	0.8532	0.8466	0.8436	0.8742	0.8638	0.8643	0.8853	0.8817	0.8799	0.8995	0.8905	0.8939
MASE	0.9945	0.9922	1.0030	1.1027	1.0649	1.1118	1.0542	1.0616	1.0565	1.0749	1.0577	1.0805
Bias	−0.0208	−0.0032	−0.0193	−0.0180	−0.0410	−0.0298	−0.0234	−0.0282	−0.0082	−0.0100	−0.0466	0.0106

This pattern is visible in Figure 6.3, which plots the WRMSSE trajectory of each model across scenarios. Throughout all four scenarios, the TFT and DeepAR-Direct consistently achieve lower WRMSSE values than the LSTM, with the gap between TFT and LSTM ranging from 0.0066 in Scenario 1 to 0.0090 in Scenario 4. The MASE follows a broadly similar trajectory, though with a notable dip in Scenario 3 relative to Scenario 2 for all three models. In Scenario 1, all three models record MASE values below 1.0, indicating performance superior to the seasonal naïve benchmark on this metric at single-store scale. The Bias values remain small and negative across most model-scenario combinations, with the exception of DeepAR-Direct in Scenario 4, which records a small positive bias of 0.0106.

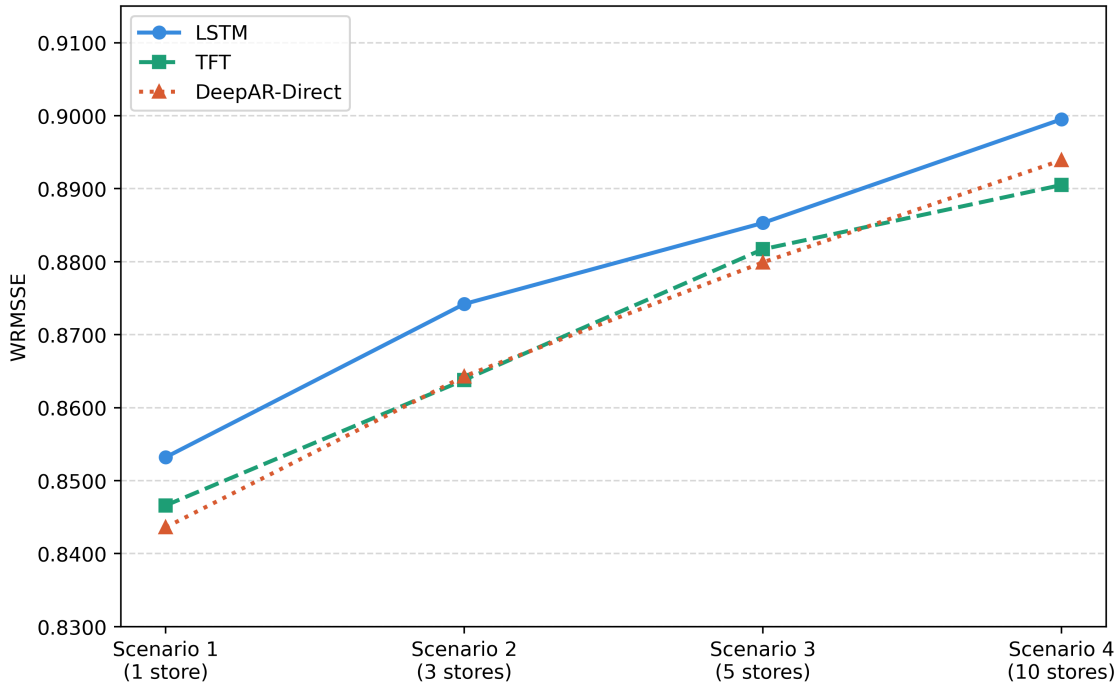


Figure 6.3: WRMSSE of each model across the four scalability scenarios.

Table 6.7 presents the computational metrics for each model across all four scenarios. The training time trajectories are illustrated in Figure 6.4, which highlights the differences in computational cost between models as the dataset grows.

Table 6.7: Computational efficiency metrics recorded for each model across the four evaluation scenarios.

Metric	Scenario 1			Scenario 2			Scenario 3			Scenario 4		
	LSTM	TFT	DeepAR-Direct	LSTM	TFT	DeepAR-Direct	LSTM	TFT	DeepAR-Direct	LSTM	TFT	DeepAR-Direct
Total training time (s)	1,060	3,203	301	2,634	6,261	501	662	3,792	1,728	11,985	18,693	2,098
Epochs to convergence	6	6	9	5	16	8	7	14	11	14	7	5
Total inference time (s)	0.0667	0.6591	0.0300	0.1974	0.5537	0.0545	0.0605	0.4469	0.1209	0.5628	3.4343	0.2560
Avg. VRAM training (MB)	15.21	6.43	3.97	15.46	23.70	4.70	15.86	85.85	4.48	4.31	21.88	4.48
Peak VRAM training (MB)	50.50	41.44	65.85	51.01	301.10	143.85	184.97	1,175.62	85.53	38.14	158.42	85.53
Peak VRAM inference (MB)	31.10	24.20	42.34	31.51	79.03	74.99	80.93	228.79	50.97	32.83	68.17	50.97
Trainable parameters	865,964	393,313	109,191	882,396	1,405,441	142,503	882,428	5,432,609	142,535	143,180	1,405,553	142,615
FLOPs training ($\times 10^6$)	4,377.87	925.36	2,613.57	4,465.95	1,415.49	8,133.80	3,572.76	1,108.17	8,113.80	5,054.30	7,077.45	4,066.90
FLOPs inference ($\times 10^6$)	1,459.29	308.45	871.92	1,488.65	471.83	2,711.27	1,190.92	369.39	3,693.89	1,684.77	2,359.15	1,355.63
Batch Size	64	64	256	64	256	512	512	512	256	64	128	256

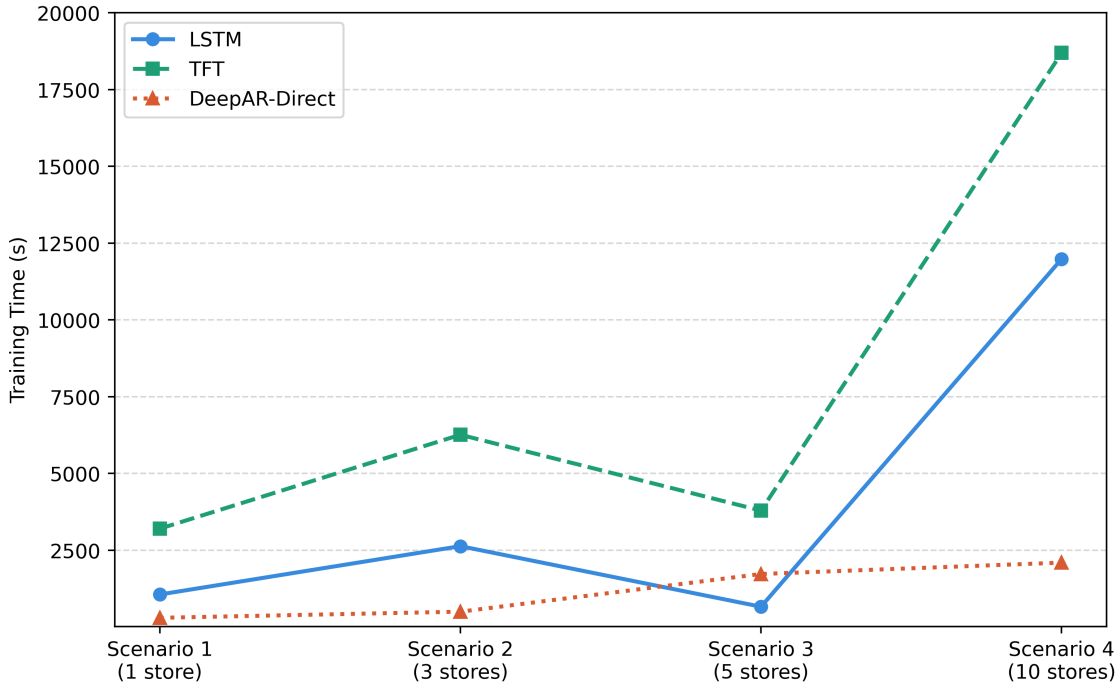


Figure 6.4: Training time of each model across the four scalability scenarios, expressed in seconds.

The training time of the TFT is consistently the highest across all scenarios, ranging from 3,203 seconds in Scenario 1 to 18,693 seconds in Scenario 4. The DeepAR-Direct is consistently the fastest to train, with times ranging from 301 seconds in Scenario 1 to 2,098 seconds in Scenario 4, a factor of approximately 2 to 13 times faster than the TFT across scenarios. The LSTM occupies an intermediate position in Scenarios 1 and 2, but its training time increases sharply to 11,985 seconds in Scenario 4, narrowing the gap with the TFT considerably. The non-monotonic behaviour visible in the training time curves reflects changes in both the batch size and the number of epochs to convergence selected by Optuna rather than a genuine reduction in computational cost per sample. This is evident in the drop in training time between Scenario 2 and Scenario 3 for both the TFT and the LSTM.

The number of epochs to convergence varies considerably across models and scenarios. The TFT requires the most epochs in Scenario 2 (16) and Scenario 3 (14), which contributes to its elevated training times in those scenarios, before converging in just 7 epochs in Scenario 4 despite the larger dataset. The LSTM converges in 5 to 7 epochs across Scenarios 1 through 3 but requires 14 epochs in Scenario 4, partially explaining the sharp increase in training time at full scale. The DeepAR-Direct shows the most stable convergence behaviour, requiring between 5 and 11 epochs across all scenarios.

Inference time follows a pattern broadly consistent with training time. The TFT requires the most time to generate forecasts across all scenarios, reaching 3.43 seconds in Scenario 4. The DeepAR-Direct is again the fastest, completing inference in 0.26 seconds in Scenario 4. The LSTM falls between the two, with an inference time of 0.56 seconds in Scenario 4.

The peak GPU VRAM consumption during training reveals a pronounced difference between

the TFT and the other two models. Whilst the LSTM and DeepAR-Direct remain below 200 MB across all scenarios, the TFT reaches 1,175.62 MB in Scenario 3 before dropping to 158.42 MB in Scenario 4. This anomalous peak in Scenario 3 coincides with the largest batch size selected for the TFT (512), which substantially increases the memory footprint of the attention mechanism during training. The average VRAM during training is considerably lower than the peak for all models, reflecting the fact that peak consumption occurs during specific operations rather than being sustained throughout the run. During inference, peak VRAM values are substantially lower than during training for all models, with the TFT reaching a maximum of 228.79 MB in Scenario 3.

The number of trainable parameters varies considerably across models and scenarios. The TFT exhibits the largest parameter counts overall, growing from 393,313 in Scenario 1 to 5,432,609 in Scenario 3 before returning to 1,405,553 in Scenario 4, a variation driven primarily by the change in hidden size selected by Optuna. The LSTM shows a similar pattern, peaking at 882,428 parameters in Scenario 3. The DeepAR-Direct maintains the smallest and most stable parameter count across all scenarios, ranging from 109,191 to 142,615, consistent with its fixed hidden size of 64 and the limited growth of the store and item embedding layers as the number of stores increases.

The raw FLOPs per batch for training and inference are reported in Table 6.7 alongside the corresponding batch size, allowing the reader to derive per-series estimates where needed. The TFT exhibits considerably higher FLOPs in Scenario 4 compared to the other scenarios, reflecting the larger hidden size of 128 selected by Optuna. The DeepAR-Direct shows a notable increase in inference FLOPs in Scenario 3, driven by the interaction between its three-layer decoder and the larger batch size of 256.

6.2.4 Discussion

The most consistent finding across all four scalability scenarios is the degradation of forecasting accuracy as the number of stores increases. This pattern is not unexpected: as the training corpus grows to encompass stores from different states and with different demand profiles, the models must generalise across a more heterogeneous portfolio, and the fixed-capacity architectures selected by Optuna are not always sufficient to absorb the additional complexity. The accuracy degradation between Scenario 1 and Scenario 4 is broadly similar across the three models, with the DeepAR-Direct showing a marginally larger absolute increase in WRMSSE than the LSTM and TFT. Whilst these differences are modest in absolute terms, they suggest that the structured input processing of the TFT, which selects and weights features through its variable selection networks, provides a degree of robustness to portfolio heterogeneity that the simpler LSTM encoder does not.

The relationship between model complexity and forecasting accuracy is not straightforward when examined across scenarios. In Scenario 1, the three models achieve broadly similar WRMSSE values, with a spread of less than 0.01 points. As the number of stores grows, the TFT and DeepAR-Direct maintain closer WRMSSE values to each other than either does to the LSTM,

suggesting that the recurrent encoder-decoder structure of the DeepAR-Direct and the attention mechanism of the TFT share a form of representational advantage over the simpler LSTM when faced with larger and more heterogeneous portfolios. This advantage, however, comes at very different computational costs for the two models, which is where the most practically significant findings of this analysis lie.

The DeepAR-Direct emerges as the most computationally efficient architecture across all scenarios by a substantial margin. Its training times are consistently the lowest, its parameter counts are the smallest and most stable, and its inference latency remains well below that of the other two models even at full ten-store scale. This efficiency is not achieved at the expense of accuracy: the DeepAR-Direct records competitive WRMSSE values throughout, and in Scenario 1 achieves the lowest WRMSSE of the three models. The architectural explanation for this efficiency lies in the combination of its encoder-decoder design and its fixed hidden size of 64 across all scenarios. The encoder-decoder structure allows the model to process the forecast horizon sequentially without requiring a large hidden representation to summarise the full input window into a single vector, as the LSTM does. The result is a model that scales: as the number of series grows, the per-series computational cost remains stable, and the total training and inference time increases primarily as a function of dataset size rather than model complexity.

The TFT presents the opposite profile. It is consistently the most expensive model to train and to deploy, and the gap between its computational cost and that of the other two models widens substantially at full scale. The attention mechanism at the core of the TFT is inherently quadratic in the sequence length, and whilst the lookback window of 28 days is short enough to keep this cost manageable in isolation, the combination of a larger hidden size, multiple gating operations, and the static enrichment pathway results in a higher per-sample cost than either of the recurrent alternatives. The anomalous peak VRAM consumption of 1,175.62 MB in Scenario 3 illustrates the memory sensitivity of the architecture: when the batch size increases to 512, the attention computation must maintain a full set of query, key, and value representations for every sample in the batch simultaneously, placing a disproportionate burden on GPU memory relative to the LSTM or DeepAR-Direct. In practical deployment terms, this memory sensitivity is a constraint, as it limits the batch sizes that can be used during training on hardware with restricted VRAM.

The accuracy advantage of the TFT over the LSTM is real but modest, and must be weighed against its substantially higher computational cost. Across all four scenarios, the TFT achieves a lower WRMSSE than the LSTM, but the margin remains modest throughout. Whether this improvement justifies a training time that is consistently between two and thirteen times longer depends entirely on the operational context. In settings where forecasting accuracy has a direct and measurable economic impact, the additional cost may be warranted. In settings where computational resources are constrained or where models must be retrained frequently, the DeepAR-Direct offers a more practical balance between accuracy and efficiency.

A further observation concerns the stability of the hyperparameter configurations selected across scenarios. The DeepAR-Direct converges to a consistent configuration, with a hidden size of 64 and three layers from Scenario 2 onwards, suggesting that this architecture reaches a stable

optimum that generalises well across scales. The TFT and LSTM, by contrast, show more variation in their selected configurations, with hidden size, batch size, and epochs to convergence fluctuating considerably between scenarios. This instability has practical implications: it means that a configuration optimised for a smaller dataset cannot be reliably transferred to a larger one, and that hyperparameter optimisation must be repeated independently for each scale of deployment. The DeepAR-Direct's configuration stability is therefore an additional argument in its favour from a deployment perspective, beyond its raw computational efficiency.

RQ2: All three DL models exhibit a consistent degradation in forecasting accuracy as the number of stores increases, reflecting the greater heterogeneity of larger portfolios. The DeepAR-Direct is the most computationally efficient architecture across all scenarios, achieving competitive accuracy with the smallest parameter counts, lowest training times, and lowest inference latency, whilst maintaining a stable hyperparameter configuration across scales. The TFT achieves the best accuracy at full scale but at a substantially higher computational cost, with training times between two and thirteen times longer than the DeepAR-Direct and pronounced memory sensitivity at large batch sizes. The LSTM is the least accurate of the three models, and its training time grows sharply at full scale, reducing its practical advantage over the TFT. These findings suggest that the choice of architecture for scalable retail demand forecasting involves a meaningful trade-off between accuracy and computational cost, and that the DeepAR-Direct offers the most favourable balance across the scenarios evaluated in this study.

6.3 Robustness Analysis

This section addresses the **third research question** by evaluating how the predictive performance of the three DL models varies across the four demand pattern classes defined by the Syntetos-Boylan framework. Performance is decomposed at series level using the MASE and Bias metrics, and complemented by an analysis of time series properties extracted with the `ete_ts`[4] library, with the aim of identifying which signal characteristics are most strongly associated with forecast difficulty within each demand pattern class.

6.3.1 Experimental Setup

The robustness analysis is conducted on the results of Scenario 4, the full ten-store dataset introduced in Section 6.2.1. With 30,490 store-item series spanning stores from three US states and covering the full heterogeneity of the M5 portfolio, Scenario 4 provides the most representative basis for drawing conclusions about model behaviour across demand patterns. A smaller scenario would preserve the same pattern distribution but reduce the statistical weight behind each class,

[4] https://franciscovmacieira.github.io/ete_ts/, accessed 6 Jun 2026.

particularly Smooth and Erratic, which together account for fewer than 3,000 series even at full scale.

The Syntetos-Boylan classification, introduced in Section 5.1, provides the framework through which robustness is examined. By partitioning the portfolio along the two dimensions of demand frequency and size variability, it groups series that present structurally similar forecasting challenges, making it possible to isolate how model performance varies with the nature of the demand signal rather than with arbitrary product or store characteristics. Each class therefore represents a qualitatively distinct test of model robustness, and the degree to which performance degrades from Smooth to Intermittent is a direct measure of how well each architecture handles the irregular and sparse demand structures that dominate food retail portfolios.

The choice of MASE as the primary accuracy measure for this analysis follows directly from the nature of the question being asked. The WRMSSE, used as the primary metric in the first research question, weights each series by its contribution to total sales volume, which is appropriate when the goal is to assess economic impact at portfolio level. Here, the goal is different: to understand how model behaviour varies across series of different structural character, regardless of their commercial weight. The MASE, computed independently for each series and aggregated without weighting across each demand class, treats a slow-moving Intermittent series and a high-volume Smooth series as equally informative observations, which is precisely what a structural analysis requires. Bias is examined alongside MASE to capture the directional dimension of errors within each class. Whilst its scale-dependence precludes direct comparison across individual series, it remains informative as an aggregate measure within each demand class, where the interest lies in the sign and relative magnitude of systematic error rather than in cross-series comparisons.

The analysis proceeds in two stages. The first examines the distribution of MASE and Bias across the four demand pattern classes for all three models, characterising how error magnitude and directional tendency shift as demand becomes sparser and more irregular. The second stage examines, within each demand class, the relationship between per-series signal characteristics and forecast error, using a set of time series properties extracted with the `ete_ts` library. These properties were selected because they collectively cover the dimensions most directly relevant to forecast difficulty in retail demand: signal volatility, predictable structure, seasonal regularity, temporal memory, and structural complexity. Crucially, they capture variation within each Syntetos-Boylan class that the ADI and CV^2 thresholds cannot resolve, since two series may share the same classification whilst differing in their underlying temporal dynamics. Five properties are therefore considered: fluctuation, which measures the proportion of large changes in the series; forecastability, which quantifies the strength of predictable structure relative to noise; seasonal strength, which captures the magnitude of the weekly seasonal component; autocorrelation relevance, which measures how rapidly the autocorrelation function decays as a proxy for temporal memory; and complexity, which estimates the structural richness of the normalised series. These properties are examined through scatter plots stratified by demand class and model, with the aim of identifying which specific signal characteristics are most strongly associated with forecast error within each pattern category.

6.3.2 Results

Figure 6.5 presents the distribution of per-series MASE across the four demand pattern classes for all three models, and Table 6.8 reports the corresponding descriptive statistics. The most immediate observation is the pronounced and consistent degradation in forecast accuracy as demand becomes sparser and more irregular. Across all three models, median MASE rises from approximately 0.74 in the Smooth class to values approaching or exceeding 1.0 in the Intermittent class, with Lumpy series occupying an intermediate position at medians around 0.85 to 0.87. The Erratic class presents a partial exception to this progression: despite its high size variability, its median MASE is slightly lower than that of Smooth series for all three models, with values of 0.701, 0.688, and 0.699 for LSTM, TFT, and DeepAR-Direct respectively. This counterintuitive result is accompanied by a wider Interquartile Range (IQR), reflecting greater dispersion rather than uniformly better performance across the class.

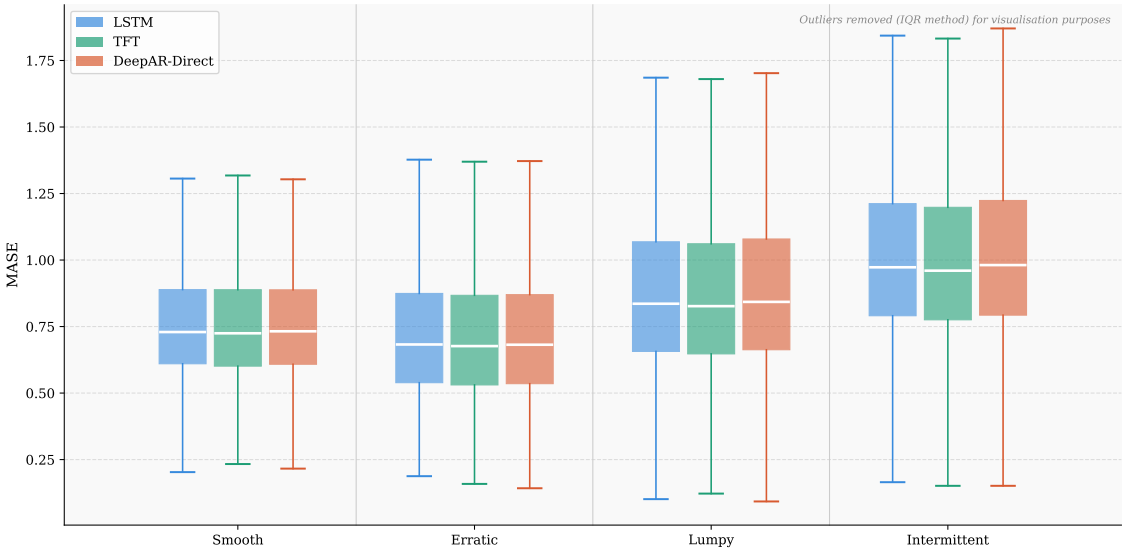


Figure 6.5: MASE distribution by demand pattern class for all three models. Outliers removed using the IQR method for visualisation purposes.

The spread of errors, as measured by the IQR, follows a pattern that is as informative as the central tendency. For Smooth and Erratic series, the IQR remains below 0.38 across all models, indicating a relatively concentrated error distribution. For Lumpy series, the IQR expands to approximately 0.47 to 0.48, and for Intermittent series it stabilises at around 0.47 across all three models. The percentage of series recording a MASE above 1.0 rises sharply across the demand classes, from below 20% in the Smooth and Erratic classes to 35% in the Lumpy class, reaching values between 47.7% and 50.2% in the Intermittent class depending on the model. Among the three architectures, the TFT records the lowest proportion of Intermittent series above this threshold at 47.7%, whilst the DeepAR-Direct records the highest at 50.2%.

Across all four demand pattern classes, the three models produce similar MASE distributions. The TFT records the lowest median in every class, with margins of at most 0.013 MASE

Table 6.8: Descriptive statistics of per-series MASE by demand pattern class and model.

Pattern	Model	Median	IQR	Q1	Q3	Min	Max	% MASE > 1
Smooth	LSTM	0.744	0.309	0.615	0.924	0.203	3.655	18.5
	TFT	0.737	0.310	0.607	0.917	0.233	3.843	17.9
	DeepAR-Direct	0.745	0.303	0.615	0.919	0.216	3.566	18.5
Erratic	LSTM	0.701	0.368	0.550	0.917	0.188	3.998	19.6
	TFT	0.688	0.376	0.539	0.915	0.159	3.965	19.2
	DeepAR-Direct	0.699	0.360	0.547	0.907	0.142	4.618	18.8
Lumpy	LSTM	0.859	0.472	0.666	1.138	0.101	24.360	35.0
	TFT	0.852	0.476	0.658	1.133	0.122	23.133	34.2
	DeepAR-Direct	0.868	0.479	0.673	1.152	0.093	27.876	35.7
Intermittent	LSTM	0.995	0.468	0.801	1.269	0.127	50.940	49.3
	TFT	0.980	0.468	0.786	1.254	0.109	36.836	47.7
	DeepAR-Direct	1.002	0.478	0.803	1.281	0.121	42.057	50.2

points over the LSTM and 0.022 points over the DeepAR-Direct within any single class. These differences are consistent in direction but narrow in magnitude, suggesting that the architectural distinctions between the three models do not translate into substantially different robustness profiles across demand pattern classes. All three architectures degrade in a broadly parallel fashion as demand becomes more irregular.

Figure 6.6 and Table 6.9 present the Bias distributions across the four demand pattern classes. The most striking feature of these results is how consistently the median Bias remains close to zero across all classes and all models. In every combination of pattern class and model, the median lies between -0.119 and 0.029, indicating that none of the three architectures develops a systematic directional tendency that is stable across the series within any given class.

What the Bias distributions do reveal, however, is a substantial difference in dispersion across demand pattern classes that is not visible in the MASE results. The IQR of Bias is widest for Erratic series, reaching 1.859 for the LSTM, 1.853 for the TFT, and 1.741 for the DeepAR-Direct, values that are approximately seven times larger than the corresponding figures for Intermittent series, where the IQR narrows to around 0.25 across all models. For Lumpy series, the IQR occupies an intermediate position at values between 0.572 and 0.587 across the three models.

A consistent pattern across all demand classes is a slight tendency for the TFT to record more negative median Bias values than the LSTM and DeepAR-Direct. In the Erratic class, the TFT median reaches -0.119, compared to -0.032 for the LSTM and -0.047 for the DeepAR-Direct. In the Lumpy class, the TFT records -0.028 against values of -0.004 and 0.026 for the LSTM and DeepAR-Direct respectively. In the Smooth and Intermittent classes, the same directional pattern holds, though the differences between models are smaller in magnitude.

Figures 6.7, 6.8, and 6.9 present scatter plots of fluctuation against per-series MASE for the LSTM, TFT, and DeepAR-Direct respectively, with each panel corresponding to one of the four

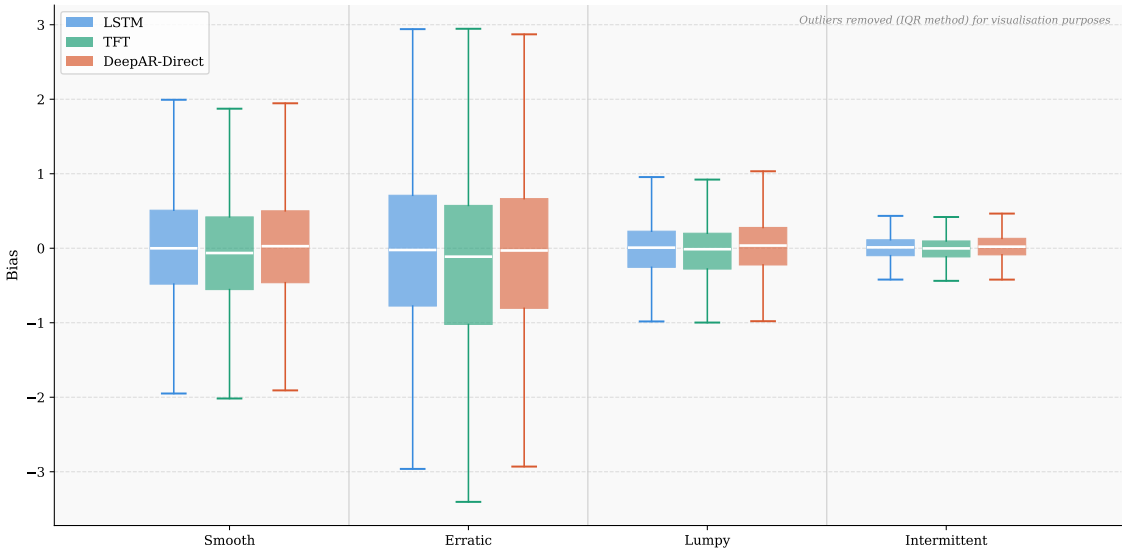


Figure 6.6: Bias distribution by demand pattern class for all three models. Outliers removed using the IQR method for visualisation purposes.

Table 6.9: Descriptive statistics of per-series Bias by demand pattern class and model.

Pattern	Model	Median	IQR	Q1	Q3	Min	Max	% Bias > 0
Smooth	LSTM	0.013	1.221	−0.564	0.656	−35.549	30.208	50.7
	TFT	−0.073	1.233	−0.713	0.520	−12.779	30.261	46.6
	DeepAR-Direct	0.029	1.236	−0.588	0.647	−15.173	31.266	51.3
Erratic	LSTM	−0.032	1.859	−1.006	0.853	−38.179	14.503	49.0
	TFT	−0.119	1.853	−1.184	0.669	−34.460	12.104	45.4
	DeepAR-Direct	−0.047	1.741	−1.017	0.724	−29.479	13.741	47.8
Lumpy	LSTM	−0.004	0.572	−0.324	0.248	−26.769	11.713	49.5
	TFT	−0.028	0.574	−0.359	0.214	−20.113	9.842	46.8
	DeepAR-Direct	0.026	0.587	−0.288	0.299	−17.069	21.261	52.8
Intermittent	LSTM	0.010	0.250	−0.124	0.126	−16.679	14.644	52.4
	TFT	−0.004	0.251	−0.139	0.112	−11.338	13.892	49.0
	DeepAR-Direct	0.022	0.257	−0.104	0.153	−18.309	16.152	55.6

demand pattern classes. The three figures are visually consistent across models, indicating that the relationship between fluctuation and forecast error does not vary substantially with the choice of architecture. Within the Smooth and Erratic classes, the scatter plots show a diffuse cloud with no discernible directional trend: series with high fluctuation do not systematically attract higher MASE values than those with low fluctuation, and the error distribution appears largely independent of this property within both classes. The Lumpy class presents a broadly similar picture, with the added complexity of a wider MASE range that reflects the greater difficulty of that class overall.

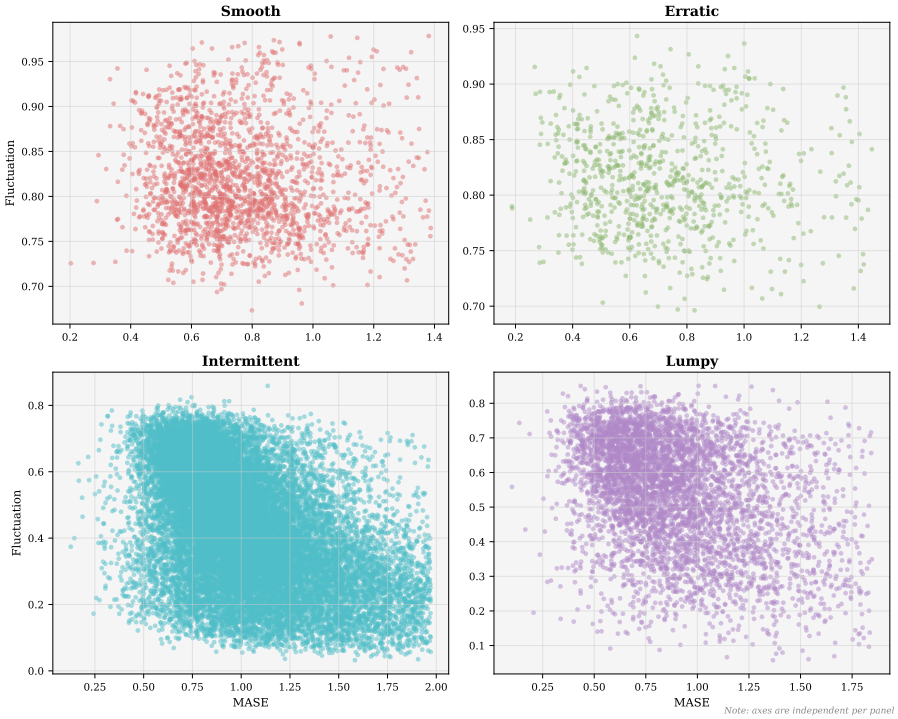


Figure 6.7: Scatter plots of fluctuation against per-series MASE for the LSTM, stratified by demand pattern class.

The Intermittent class tells a different story. Across all three models, the scatter plots reveal a characteristic asymmetric shape in which the density of points is highest at low fluctuation values and low to moderate MASE, whilst the tail of high-error series extends predominantly into the region of lower fluctuation. Series with fluctuation values above approximately 0.6 are almost exclusively concentrated in the MASE range below 1.0, whilst the proportion of series with MASE above 1.0 increases progressively as fluctuation decreases. This pattern is consistent across the LSTM, TFT, and DeepAR-Direct. The scatter plots for the remaining time series properties examined, namely forecastability, seasonal strength, autocorrelation relevance, and complexity, are presented in Appendix A for completeness.

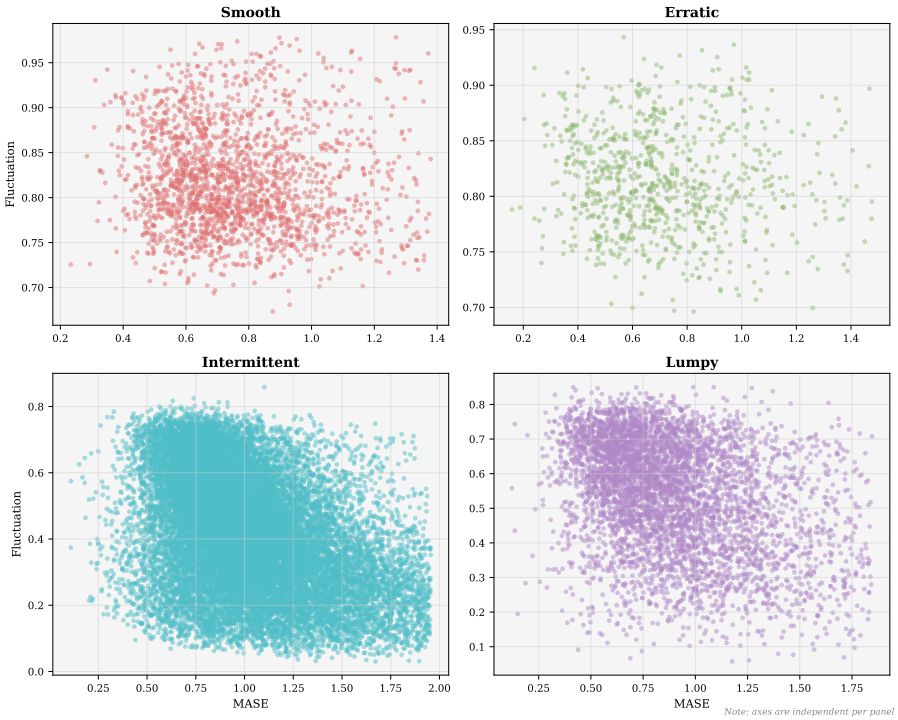


Figure 6.8: Scatter plots of fluctuation against per-series MASE for the TFT, stratified by demand pattern class.

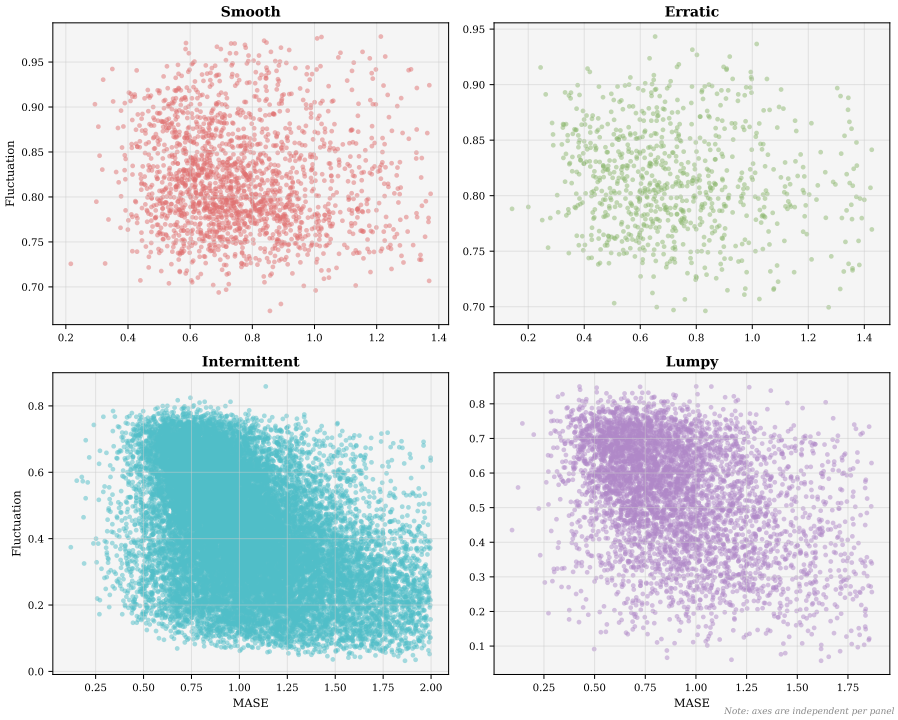


Figure 6.9: Scatter plots of fluctuation against per-series MASE for the DeepAR-Direct, stratified by demand pattern class.

6.3.3 Discussion

The results establish a clear and consistent hierarchy of difficulty across the four demand pattern classes that holds regardless of the architecture being evaluated. Smooth and Erratic series, despite their structural differences in size variability, present a tractable forecasting environment for all three models, with median MASE values well below 1.0 and fewer than one in five series exceeding that threshold. The more consequential finding is the sharp deterioration in performance at the Lumpy and Intermittent classes, where the combination of sparse demand occurrence and irregular magnitudes substantially reduces the predictive signal available to the recurrent encoders. That this degradation is nearly identical across the three architectures is itself a important result: it suggests that the robustness ceiling encountered in the Intermittent class is not an architectural limitation of any particular model but a property of the demand signal itself. When demand occurs infrequently and in irregular quantities, the 28-day lookback window carries limited exploitable structure, and the capacity advantages of the TFT's attention mechanism or the DeepAR-Direct's encoder-decoder design provide little additional leverage over the simpler LSTM encoder.

The Erratic class warrants particular attention. Its median MASE is marginally lower than that of Smooth series across all three models, a result that appears counterintuitive given that Erratic series are defined by high size variability. The explanation lies in the distinction between what the Syntetos-Boylan classification captures and what drives MASE specifically. The ADI threshold that separates Smooth from Erratic series concerns demand frequency, not temporal predictability: Erratic series occur as regularly as Smooth ones, which means the recurrent encoders receive a dense observation signal at every timestep. The high CV^2 of Erratic series introduces uncertainty in the magnitude of individual forecasts, but the regularity of demand occurrence ensures that the models are never confronted with the structural sparsity that characterises Intermittent and Lumpy series. The wider IQR of Erratic MASE relative to Smooth reflects this magnitude uncertainty without shifting the central tendency upward.

The Bias results reveal a dimension of model behaviour that the MASE analysis does not capture. The near-zero median Bias across all classes and all models confirms that none of the three architectures develops a stable directional tendency at the class level, a consequence of the Tweedie training objective, which does not penalise overestimation and underestimation asymmetrically and therefore does not drive the optimisation towards either direction systematically. The more operationally relevant finding is the pronounced difference in Bias dispersion across classes. The IQR of Bias for Erratic series is approximately seven times wider than for Intermittent series, reaching values above 1.8 for the LSTM and TFT. This asymmetry reflects the structural reality of each class: for Intermittent series, the prevalence of zero demand constrains the range of possible signed errors, and the models converge towards low-magnitude predictions that are similarly calibrated regardless of the true demand level. For Erratic series, the high variability of non-zero demand sizes means that individual series can attract large positive or negative errors, and the wide Bias IQR reflects this sensitivity to magnitude uncertainty rather than a systematic directional failure. The consistently more negative median Bias of the TFT across all demand classes,

most pronounced in the Erratic class at -0.119, is consistent with its portfolio-level underprediction tendency and suggests that this tendency is not confined to a specific demand structure but is a general characteristic of the model's behaviour that manifests with varying intensity depending on the size variability of the series.

The scatter plots of fluctuation against per-series MASE provide the most granular insight into the drivers of forecast difficulty within individual demand classes. For Smooth, Erratic, and Lumpy series, fluctuation shows no consistent directional relationship with MASE, indicating that within these classes the volatility of the demand signal does not systematically amplify forecast error. The Intermittent class is the exception. The asymmetric distribution of points, in which high-fluctuation series cluster at low MASE whilst low-fluctuation series extend into the high-error tail, holds consistently across all three models and indicates that within the Intermittent class, the volatility of the demand signal is a meaningful contributor to forecast difficulty that operates independently of architectural differences. This finding refines the picture provided by the Syntetos-Boylan classification alone: two Intermittent series that share the same ADI and CV^2 can differ substantially in their forecastability depending on the magnitude of their step changes, and that difference is reflected in the error distributions of all three models in the same way. The remaining time series properties examined, forecastability, seasonal strength, autocorrelation relevance, and complexity, did not reveal consistent directional relationships with MASE within any demand class across the three models, as shown in Appendix A, suggesting that fluctuation is the property most strongly associated with within-class variation in forecast difficulty for the Intermittent demand pattern.

RQ3: All three DL models exhibit a consistent and parallel degradation in forecast accuracy as demand becomes sparser and more irregular, with median MASE rising from approximately 0.74 in the Smooth class to values at or above 1.0 in the Intermittent class. The similarity of this degradation across architectures indicates that the robustness ceiling encountered in the Intermittent class reflects the informational constraints of sparse demand signals rather than the limitations of any individual model. None of the three architectures develops a systematic directional bias within any demand class, though the TFT displays a consistently more negative median Bias across all classes, in line with its portfolio-level underprediction tendency. Within the Intermittent class, which accounts for 73.3% of the portfolio, fluctuation emerges as the time series property most strongly associated with forecast difficulty, with low-fluctuation series contributing predominantly to the high-error tail across all three models.

Chapter 7

Conclusions and Future Work

This chapter presents the conclusions drawn from the empirical work conducted in this dissertation and outlines directions for future research. Section 7.1 synthesises the main findings across the three research questions. Section 7.2 identifies the most promising directions for extending this study.

7.1 Conclusions

This dissertation investigated whether advanced DL architectures, trained on large-scale real operational data, can deliver practical forecasting value in the food retail sector. The empirical work was conducted using the M5 Walmart dataset and evaluated three global architectures, LSTM, TFT, and DeepAR-Direct, across three complementary dimensions: predictive accuracy, computational scalability, and robustness to heterogeneous demand patterns.

The accuracy results confirm that global DL architectures, each trained across thousands of store-item series simultaneously, consistently outperform classical statistical baselines and global ML methods on real operational data. The gains are modest in absolute terms but uniform across the three architectures. The proximity of the proposed models to the M5 competition winners, achieved without price information, promotional flags, or ensemble strategies, suggests that the core modelling decisions adopted here capture a substantial portion of the attainable accuracy from sales history alone. The divergence between WRMSSE and MASE rankings reveals, however, that these gains are not uniformly distributed across the portfolio. The advantage of the DL models is concentrated in higher-volume series, whilst the sparse and irregular demand of the Intermittent majority provides limited signal for the recurrent encoders to exploit.

The scalability analysis reveals that the architecture selection problem in retail forecasting extends well beyond predictive accuracy. The three models degrade in accuracy as the portfolio grows, but the rate and cost of that degradation differ, and those differences can have direct operational consequences. The DeepAR-Direct demonstrates that competitive accuracy and computational efficiency are not mutually exclusive: its encoder-decoder design scales to larger portfolios without the memory sensitivity or training time volatility that constrain the TFT in larger

scenarios. The TFT's accuracy advantage at full scale is real but must be weighed against a higher computational cost. The LSTM is the least accurate of the three models, whilst occupying an intermediate position on computational cost, faster than the TFT but falling short of the DeepAR-Direct on efficiency. These findings reframe architecture selection as a trade-off problem rather than a purely technical one, where the right choice depends as much on the deployment context as on the accuracy figures.

The robustness analysis delivers perhaps the most consequential finding of the three. The parallel degradation of all three architectures as demand becomes sparser and more irregular points to a ceiling that appears to be set not by the models themselves but by the informational content of the demand signal. In a portfolio dominated by Intermittent series, the 28-day lookback window carries limited exploitable structure, and the capacity advantages of attention mechanisms or encoder-decoder designs provide little additional leverage over a simpler recurrent encoder.

The three research questions, taken together, demonstrate that global DL architectures trained on real operational data are a viable and competitive approach to demand forecasting in food retail.

7.2 Future Work

The findings of this dissertation open several directions for future research, each building on the empirical foundations established here.

The most direct extension concerns the incorporation of exogenous variables into the model inputs. The current study deliberately limits the input representation to sales history and categorical identifiers, a decision motivated by the goal of isolating the contribution of the core architectural choices from the effect of additional features. The gap between the models proposed here and the two M5 competition participants is at least partially attributable to the absence of price information, promotional flags, SNAP purchase indicators, and calendar events, all of which are available in the M5 dataset and were exploited by both competition solutions. Incorporating these variables would provide a more complete assessment of what these architectures can achieve under realistic operational conditions. It would also allow a direct quantification of the marginal contribution of exogenous information relative to the sales-history-only baseline established in this study.

The lookback window length was fixed at 28 days across all experiments, matching the forecast horizon and aligning with the setup of the third-placed M5 solution. Whilst this choice is practically motivated, it was not empirically optimised, and there is no a priori reason to expect that 28 days represents the optimal historical context for all demand pattern classes. For Smooth and Erratic series, where demand is dense and recent history is informative, a shorter window may be sufficient. For Intermittent and Lumpy series, a longer window might provide additional context about the frequency and magnitude of past demand events. Treating the lookback length as a hyperparameter subject to optimisation, rather than a fixed design constant, would allow each architecture to adapt its temporal receptive field to the structure of the series it is trained on.

The evaluation in this dissertation is conducted exclusively at a forecast horizon of 28 days, inherited from the M5 competition setup. Extending the analysis to shorter and longer horizons

would establish whether the relative performance of the three architectures is stable across temporal scales or whether certain designs are better suited to specific planning contexts. Furthermore, all three models adopt a direct multi-step forecasting strategy that produces all horizon steps simultaneously. Whether this approach scales favourably to substantially longer horizons, or whether alternative decoding strategies become preferable as the horizon grows, remains an open question.

The three models produce point forecasts, providing a single expected demand value for each store-item-day combination without any quantification of uncertainty. In a retail inventory context, the distribution of forecast errors is as operationally relevant as the central estimate: setting safety stock levels, managing perishable products, and planning promotional inventory all require some characterisation of forecast uncertainty. Extending the architectures to produce probabilistic outputs would therefore open a complementary line of evaluation, particularly for the Intermittent and Lumpy classes where the high variance of per-series errors suggests that uncertainty quantification may carry operational value.

The computational cost of the TFT raises a practical question about its viability in resource-constrained deployment environments. Investigating strategies to reduce its computational footprint whilst preserving its accuracy advantage would be a natural extension of the scalability analysis conducted here. The practical deployability of transformer-based forecasting models in retail settings depends, to a large extent, on whether that trade-off can be made more favourable.

Finally, all experiments are conducted on a single dataset from a single retailer operating in the US. The generalisability of the findings to other retail contexts, different geographies, or different store formats remains an open question. Evaluating whether the models transfer effectively to a different retailer's portfolio would provide evidence on the broader applicability of the conclusions drawn here. Alternatively, training from scratch on a different dataset would establish whether the relative performance of the three architectures is consistent across operational contexts or specific to the characteristics of the M5 portfolio.

References

[1] Mahdi Abolghasemi et al. “Demand forecasting in supply chain: The impact of demand volatility in the presence of promotion”. en. In: *Computers & Industrial Engineering* 142 (Apr. 2020), p. 106380. ISSN: 03608352. DOI: 10.1016/j.cie.2020.106380.

[2] Y. Bengio, P. Simard, and P. Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166. DOI: 10.1109/72.279181.

[3] Lindsay R. Berry, Paul Helman, and Mike West. “Probabilistic forecasting of heterogeneous consumer transaction–sales time series”. en. In: *International Journal of Forecasting* 36.2 (Apr. 2020), pp. 552–569. ISSN: 01692070. DOI: 10.1016/j.ijforecast.2019.07.007.

[4] Alexandra Birkmaier, Adhurim Imeri, and Gerald Reiner. “Improving supply chain planning for perishable food: data-driven implications for waste prevention”. en. In: *Journal of Business Economics* 94.6 (Aug. 2024), pp. 1–36. ISSN: 0044-2372, 1861-8928. DOI: 10.1007/s11573-024-01191-x.

[5] Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne. “Machine Learning Strategies for Time Series Forecasting”. In: *Business Intelligence: Second European Summer School, eBISS 2012, Brussels, Belgium, July 15-21, 2012, Tutorial Lectures*. Ed. by Marie-Aude Aufaure and Esteban Zimányi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 62–77. ISBN: 978-3-642-36318-4. DOI: 10.1007/978-3-642-36318-4_3.

[6] G.E.P. Box et al. *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics. Wiley, 2015. ISBN: 9781118674925.

[7] R. Caetano, J.M. Oliveira, and P. Ramos. “Transformer-Based Models for Probabilistic Time Series Forecasting with Explanatory Variables”. In: *Mathematics* 13.5 (2025). DOI: 10.3390/math13050814.

[8] Vitor Cerqueira, Luis Torgo, and Igor Mozetič. “Evaluating time series forecasting models: an empirical study on performance estimation methods”. en. In: *Machine Learning* 109.11 (Nov. 2020), pp. 1997–2028. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/s10994-020-05910-7. (Visited on 06/10/2026).

[9] Ernest Chiew and Shin Siang Choong. “A solution for M5 Forecasting - Uncertainty: Hybrid gradient boosting and autoregressive recurrent neural network for quantile estimation”. In: *Special Issue: M5 competition* 38.4 (Oct. 2022), pp. 1442–1447. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2022.01.009.

[10] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)”. In: *arXiv: Learning* (2015).

[11] Kaoutar Douaioui et al. “Machine Learning and Deep Learning Models for Demand Forecasting in Supply Chain Management: A Critical Review”. en. In: *Applied System Innovation* 7.5 (Sept. 2024), p. 93. ISSN: 2571-5577. DOI: 10.3390/asi7050093.

[12] Md. Iftekharul Alam Efat et al. “Deep-learning model using hybrid adaptive trend estimated series for modelling and forecasting sales”. en. In: *Annals of Operations Research* 339.1-2 (Aug. 2024), pp. 297–328. ISSN: 0254-5330, 1572-9338. DOI: 10.1007/s10479-022-04838-6.

[13] Vijay Ekambaram et al. “Attention based Multi-Modal New Product Sales Time-series Forecasting”. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’20. event-place: Virtual Event, CA, USA. New York, NY, USA: Association for Computing Machinery, 2020, pp. 3110–3118. ISBN: 978-1-4503-7998-4. DOI: 10.1145/3394486.3403362.

[14] Vijay Ekambaram et al. “TSMixer: Lightweight MLP-Mixer Model for Multivariate Time Series Forecasting”. In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD ’23. event-place: Long Beach, CA, USA. New York, NY, USA: Association for Computing Machinery, 2023, pp. 459–469. ISBN: 979-8-4007-0103-0. DOI: 10.1145/3580305.3599533.

[15] Yasaman Ensafi et al. “Time-series forecasting of seasonal items sales using machine learning – A comparative analysis”. In: *International Journal of Information Management Data Insights* 2.1 (Apr. 2022), p. 100058. ISSN: 2667-0968. DOI: 10.1016/j.jjimei.2022.100058.

[16] Taha Falatouri et al. “Predictive Analytics for Demand Forecasting – A Comparison of SARIMA and LSTM in Retail SCM”. In: *3rd International Conference on Industry 4.0 and Smart Manufacturing* 200 (Jan. 2022), pp. 993–1003. ISSN: 1877-0509. DOI: 10.1016/j.procs.2022.01.298.

[17] Jili Fan et al. “Optimizing Attention in a Transformer for Multihorizon, Multienergy Load Forecasting in Integrated Energy Systems”. en. In: *IEEE Transactions on Industrial Informatics* 20.8 (Aug. 2024), pp. 10238–10248. ISSN: 1551-3203, 1941-0050. DOI: 10.1109/TII.2024.3392278.

[18] Andréia B.A. Ferreira, Jonatas B. Leite, and Denis H.P. Salvadeo. “Power substation load forecasting using interpretable transformer-based temporal fusion neural networks”. In: *Electric Power Systems Research* 238 (Jan. 2025), p. 111169. ISSN: 0378-7796. DOI: 10.1016/j.epsr.2024.111169.

[19] Robert Fildes, Shaohui Ma, and Stephan Kolassa. “Retail forecasting: Research and practice”. en. In: *International Journal of Forecasting* 38.4 (Oct. 2022), pp. 1283–1318. ISSN: 01692070. DOI: 10.1016/j.ijforecast.2019.06.004.

[20] E. Giacomazzi, F. Haag, and K. Hopf. “Short-Term Electricity Load Forecasting Using the Temporal Fusion Transformer: Effect of Grid Hierarchies and Data Sources”. In: 2023, pp. 353–360. DOI: 10.1145/3575813.3597345.

[21] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep Sparse Rectifier Neural Networks”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Nov. 2011, pp. 315–323.

[22] S. Goel and R. Bajpai. "Impact of Uncertainty in the Input Variables and Model Parameters on Predictions of a Long Short Term Memory (LSTM) Based Sales Forecasting Model". In: *Machine Learning and Knowledge Extraction* 2.3 (2020). DOI: 10.3390/make2030014.

[23] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2016. ISBN: 9780262337373.

[24] Cheng Guo and Felix Berkhahn. *Entity Embeddings of Categorical Variables*. 2016. arXiv: 1604.06737 [cs.LG].

[25] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. "Recurrent Neural Networks for Time Series Forecasting: Current status and future directions". In: *International Journal of Forecasting* 37.1 (2021), pp. 388–427. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2020.06.008.

[26] Geoffrey Hinton, Alex Krizhevsky, and Ilya Sutskever. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems* (Jan. 2012), pp. 1097–1105. DOI: 10.1145/3065386.

[27] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[28] S.-C. Hsieh. "Tourism demand forecasting based on an lstm network and its variants". In: *Algorithms* 14.8 (2021). DOI: 10.3390/a14080243.

[29] Jakob Huber and Heiner Stuckenschmidt. "Daily retail demand forecasting using machine learning with emphasis on calendric special days". en. In: *International Journal of Forecasting* 36.4 (Oct. 2020), pp. 1420–1438. ISSN: 01692070. DOI: 10.1016/j.ijforecast.2020.02.005.

[30] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018. ISBN: 9780987507112.

[31] YeonJun In and Jae-Yoon Jung. "Simple averaging of direct and recursive forecasts via partial pooling using machine learning". In: *International Journal of Forecasting* 38.4 (2022). Special Issue: M5 competition, pp. 1386–1399. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2021.11.007.

[32] Tim Januschowski et al. "Criteria for classifying forecasting methods". In: *International Journal of Forecasting* 36.1 (2020). M4 Competition, pp. 167–177. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2019.05.008.

[33] Yunho Jeon and Sihyeon Seong. "Robust recurrent network model for intermittent time-series forecasting". In: *Special Issue: M5 competition* 38.4 (Oct. 2022), pp. 1415–1425. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2021.07.004.

[34] Reuben Varghese Joseph et al. "A hybrid deep learning framework with CNN and Bi-directional LSTM for store item demand forecasting". In: *Computers and Electrical Engineering* 103 (Oct. 2022), p. 108358. ISSN: 0045-7906. DOI: 10.1016/j.compeleceng.2022.108358.

[35] K. Honjo, X. Zhou, and S. Shimizu. "CNN-GRU Based Deep Learning Model for Demand Forecast in Retail Industry". In: *2022 International Joint Conference on Neural Networks (IJCNN)*. Journal Abbreviation: 2022 International Joint Conference on Neural Networks (IJCNN). July 2022, pp. 1–8. ISBN: 2161-4407. DOI: 10.1109/IJCNN55064.2022.9892599.

[36] Jared Kaplan et al. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG]. URL: https://arxiv.org/abs/2001.08361.

[37] Shachar Kaufman et al. "Leakage in data mining: Formulation, detection, and avoidance". In: *ACM Trans. Knowl. Discov. Data* 6.4 (Dec. 2012). ISSN: 1556-4681. DOI: 10.1145/2382577.2382579.

[38] Diederik Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations* (Dec. 2014).

[39] Stephan Kolassa. "Evaluating predictive count data distributions in retail sales forecasting". In: *International Journal of Forecasting* 32.3 (2016), pp. 788–803. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2015.12.004.

[40] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (May 2015), pp. 436–444. DOI: 10.1038/nature14539.

[41] Dan Li et al. "Probabilistic forecasting method for mid-term hourly load time series based on an improved temporal fusion transformer model". In: *International Journal of Electrical Power & Energy Systems* 146 (Mar. 2023), p. 108743. ISSN: 0142-0615. DOI: 10.1016/j.ijepes.2022.108743.

[42] J. Li et al. "DeepAR-Attention probabilistic prediction for stock price series". In: *Neural Computing and Applications* 36.25 (2024), pp. 15389–15406. DOI: 10.1007/s00521-024-09916-3.

[43] Q. Li and M. Yu. "Achieving Sales Forecasting with Higher Accuracy and Efficiency: A New Model Based on Modified Transformer". In: *Journal of Theoretical and Applied Electronic Commerce Research* 18.4 (2023), pp. 1990–2006. DOI: 10.3390/jtaer18040100.

[44] Bryan Lim et al. "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting". In: *International Journal of Forecasting* 37.4 (Oct. 2021), pp. 1748–1764. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2021.03.012.

[45] H. Lim, K. Chung, and S. Lee. "Probabilistic Forecasting for Demand of a Bike-Sharing Service Using a Deep-Learning Approach". In: *Sustainability (Switzerland)* 14.23 (2022). DOI: 10.3390/su142315889.

[46] B. Liu et al. "Regional differences in China's electric vehicle sales forecasting: Under supply-demand policy scenarios". In: *Energy Policy* 177 (2023). DOI: 10.1016/j.enpol.2023.113554.

[47] Rong Liu and Vinay Vakharia. "Optimizing Supply Chain Management Through BO-CNN-LSTM for Demand Forecasting and Inventory Management". In: *Journal of Organizational and End User Computing* 36.1 (Aug. 2024). ISSN: 1546-2234. DOI: 10.4018/JOEUC.335591.

[48] Monte Lunacek et al. "A data-driven operational model for traffic at the Dallas Fort Worth International Airport". In: *Journal of Air Transport Management* 94 (July 2021), p. 102061. ISSN: 0969-6997. DOI: 10.1016/j.jairtraman.2021.102061.

[49] M. A. Rafi et al. "A Hybrid Temporal Convolutional Network and Transformer Model for Accurate and Scalable Sales Forecasting". In: *IEEE Open Journal of the Computer Society* 6 (2025), pp. 380–391. ISSN: 2644-1268. DOI: 10.1109/OJCS.2025.3538579.

[50] Shaohui Ma and Robert Fildes. "Retail sales forecasting with meta-learning". In: *European Journal of Operational Research* 288.1 (Jan. 2021), pp. 111–128. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2020.05.038.

[51] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. "M5 accuracy competition: Results, findings, and conclusions". en. In: *International Journal of Forecasting* 38.4 (Oct. 2022), pp. 1346–1364. ISSN: 01692070. DOI: 10.1016/j.ijforecast.2021.11.013.

[52] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. "The M5 competition: Background, organization, and implementation". en. In: *International Journal of Forecasting* 38.4 (Oct. 2022), pp. 1325–1336. ISSN: 01692070. DOI: 10.1016/j.ijforecast.2021.07.007.

[53] Spyros Makridakis, Steven Wheelwright, and {Rob J} Hyndman. *Forecasting: Methods and Applications, 3rd Ed*. English. John Wiley & Sons, 1997. ISBN: 0-471-53233-9.

[54] Vera Lucia Miguéis et al. "Reducing fresh fish waste while ensuring availability: Demand forecast using censored data and machine learning". In: *Journal of Cleaner Production* 359 (July 2022), p. 131852. ISSN: 0959-6526. DOI: 10.1016/j.jclepro.2022.131852.

[55] N. Mughees et al. "Deep sequence to sequence Bi-LSTM neural networks for day-ahead peak load forecasting". In: *Expert Systems with Applications* 175 (2021). DOI: 10.1016/j.eswa.2021.114844.

[56] M. Nasseri et al. "Applying Machine Learning in Retail Demand Prediction—A Comparison of Tree-Based Ensembles and Long Short-Term Memory-Based Deep Learning". In: *Applied Sciences (Switzerland)* 13.19 (2023). DOI: 10.3390/app131911112.

[57] J.M. Oliveira and P. Ramos. "Cross-Learning-Based Sales Forecasting Using Deep Learning via Partial Pooling from Multi-level Data". In: vol. 1826 CCIS. 2023, pp. 279–290. DOI: 10.1007/978-3-031-34204-2_24.

[58] S. Öztürk et al. "The derived demand for advertising expenses and implications on sustainability: a comparative study using deep learning and traditional machine learning methods". In: *Annals of Operations Research* 339.1-2 (2024), pp. 131–161. DOI: 10.1007/s10479-021-04429-x.

[59] P. Kaunchi et al. "Future Sales Prediction For Indian Products Using Convolutional Neural Network-Long Short Term Memory". In: *2021 2nd Global Conference for Advancement in Technology (GCAT)*. Journal Abbreviation: 2021 2nd Global Conference for Advancement in Technology (GCAT). Oct. 2021, pp. 1–5. DOI: 10.1109/GCAT52182.2021.9587668.

[60] P. S and P. D. S. "A Hybrid Demand Forecasting for Intermittent Demand Patterns using Machine Learning Techniques". In: *2022 1st International Conference on Computational Science and Technology (ICCST)*. Journal Abbreviation: 2022 1st International Conference on Computational Science and Technology (ICCST). Nov. 2022, pp. 557–561. DOI: 10.1109/ICCST55948.2022.10040407.

[61] Andrei Paleyes, Raoul-Gabriel Urma, and Neil D. Lawrence. "Challenges in Deploying Machine Learning: A Survey of Case Studies". en. In: *ACM Computing Surveys* 55.6 (July 2023), pp. 1–29. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3533378.

[62] Julian Parfitt, Mark Barthel, and Sarah Macnaughton. "Food waste within food supply chains: quantification and potential for change to 2050". en. In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 365.1554 (Sept. 2010), pp. 3065–3081. ISSN: 0962-8436, 1471-2970. DOI: 10.1098/rstb.2010.0126.

[63] S. Punia et al. "Deep learning with long short-term memory networks and random forests for demand forecasting in multi-channel retail". In: *International Journal of Production Research* 58.16 (2020), pp. 4964–4979. DOI: 10.1080/00207543.2020.1735666.

[64] P. Ramos and J.M. Oliveira. "Robust Sales forecasting Using Deep Learning with Static and Dynamic Covariates". In: *Applied System Innovation* 6.5 (2023). DOI: 10.3390/asi6050085.

[65] Cristof Rojas and Adam Jatowt. "Transformer-based probabilistic forecasting of daily hotel demand using web search behavior". In: *Knowledge-Based Systems* 310 (Feb. 2025), p. 112966. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2025.112966.

[66] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747 [cs.LG].

[67] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0.

[68] S. S. J. Nithin et al. "Retail Demand Forecasting using CNN-LSTM Model". In: *2022 International Conference on Electronics and Renewable Systems (ICEARS)*. Journal Abbreviation: 2022 International Conference on Electronics and Renewable Systems (ICEARS). Mar. 2022, pp. 1751–1756. DOI: 10.1109/ICEARS53579.2022.9752283.

[69] A. Salamanis et al. "LSTM-Based Deep Learning Models for Long-Term Tourism Demand Forecasting". In: *Electronics (Switzerland)* 11.22 (2022). DOI: 10.3390/electronics11223681.

[70] David Salinas et al. "DeepAR: Probabilistic forecasting with autoregressive recurrent networks". In: *International Journal of Forecasting* 36.3 (July 2020), pp. 1181–1191. ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2019.07.001.

[71] A. Schmidt, M.W.U. Kabir, and M.T. Hoque. "Machine Learning Based Restaurant Sales Forecasting". In: *Machine Learning and Knowledge Extraction* 4.1 (2022), pp. 105–130. DOI: 10.3390/make4010006.

[72] D. Sculley et al. "Hidden technical debt in Machine learning systems". In: *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 2503–2511.

[73] Eulalia Skawińska and Romuald I. Zalewski. "Activities of Food Retail Companies in Poland during the COVID-19 Pandemic in the Context of Food Security". In: *Sustainability* 13.13 (2021). ISSN: 2071-1050. DOI: 10.3390/su13137323.

[74] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958.

[75] A A Syntetos, J E Boylan, and J D Croston. "On the categorization of demand patterns". In: *Journal of the Operational Research Society* 56.5 (2005), pp. 495–503. DOI: 10.1057/palgrave.jors.2601841. URL: https://doi.org/10.1057/palgrave.jors.2601841.

[76] Aris A. Syntetos et al. "Supply chain forecasting: Theory, practice, their gap and the future". en. In: *European Journal of Operational Research* 252.1 (July 2016), pp. 1–26. ISSN: 03772217. DOI: 10.1016/j.ejor.2015.11.010.

[77] Leonard J. Tashman. "Out-of-sample tests of forecasting accuracy: an analysis and review". In: *International Journal of Forecasting* 16.4 (2000). The M3- Competition, pp. 437–450. ISSN: 0169-2070. DOI: https://doi.org/10.1016/S0169-2070(00)00065-0.

[78] Ewen Cameron David Todd and Dima Faour-Klingbeil. "Impact of Food Waste on Society, Specifically at Retail and Foodservice Levels in Developed and Developing Countries". en. In: *Foods* 13.13 (July 2024), p. 2098. ISSN: 2304-8158. DOI: 10.3390/foods13132098.

[79] Chih-Hsuan Wang. "Considering economic indicators and dynamic channel interactions to conduct sales forecasting for retail sectors". In: *Computers & Industrial Engineering* 165 (Mar. 2022), p. 107965. ISSN: 0360-8352. DOI: 10.1016/j.cie.2022.107965.

[80] Zhenghong Wang et al. "Spatiotemporal Fusion Transformer for large-scale traffic forecasting". In: *Information Fusion* 107 (July 2024), p. 102293. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2024.102293.

[81] Haixu Wu et al. *Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting*. en. arXiv:2106.13008 [cs]. Jan. 2022. DOI: 10.48550/arXiv.2106.13008.

[82] X. Zhang et al. "Enhancing Time Series Product Demand Forecasting With Hybrid Attention-Based Deep Learning Models". In: *IEEE Access* 12 (2024), pp. 190079–190091. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3516697.

[83] Y. Liao and C. Liang. "A Temperature Time Series Forecasting Model Based on DeepAR". In: *2021 7th International Conference on Computer and Communications (ICCC)*. Journal Abbreviation: 2021 7th International Conference on Computer and Communications (ICCC). Dec. 2021, pp. 1588–1593. DOI: 10.1109/ICCC54389.2021.9674623.

[84] Ailing Zeng et al. "Are transformers effective for time series forecasting?" In: *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*. AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN: 978-1-57735-880-0. DOI: 10.1609/aaai.v37i9.26317.

Appendix A

Scatter Plots of Time Series Properties Against Per-Series MASE by Demand Pattern Class

This appendix presents scatter plots of four time series properties extracted with the `ete_ts` library against per-series MASE, stratified by demand pattern class, for each of the three models evaluated in this study.

A.1 Forecastability

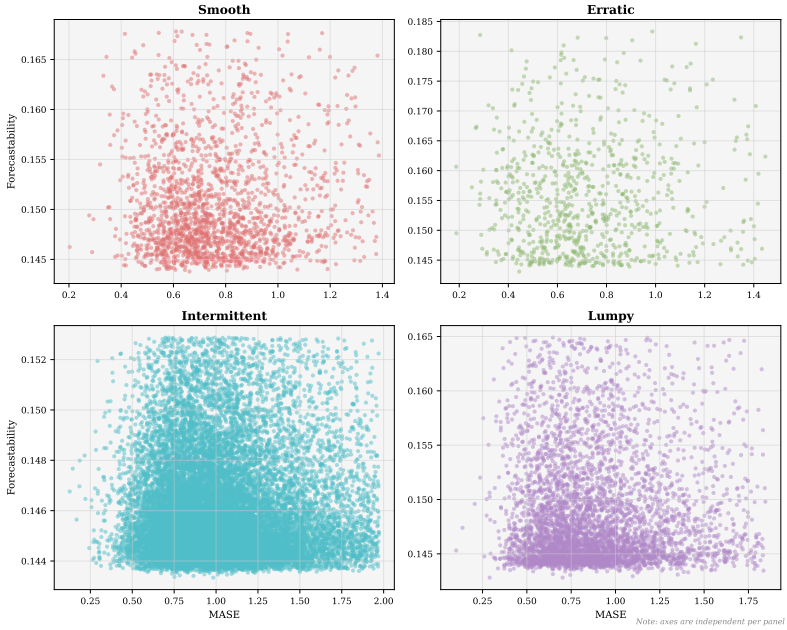


Figure A.1: Scatter plots of forecastability against per-series MASE for the LSTM, stratified by demand pattern class.

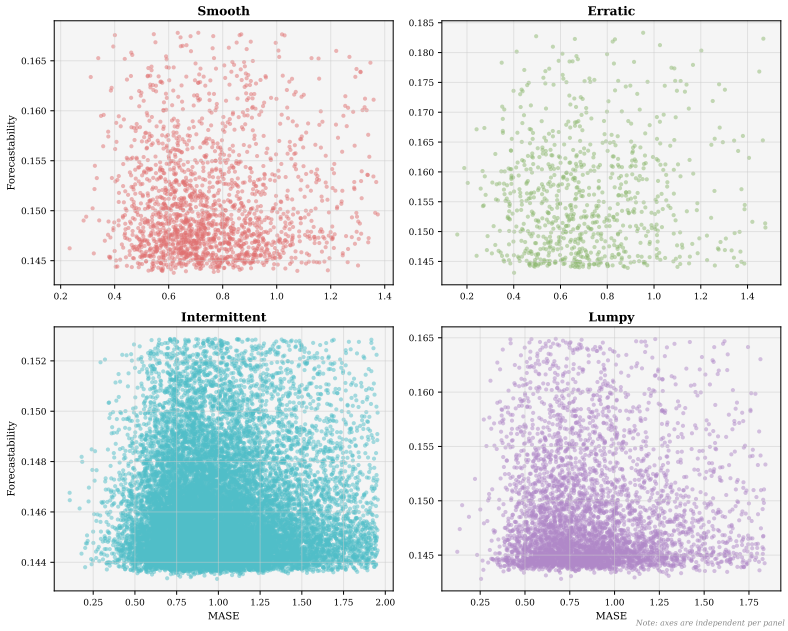


Figure A.2: Scatter plots of forecastability against per-series MASE for the TFT, stratified by demand pattern class.

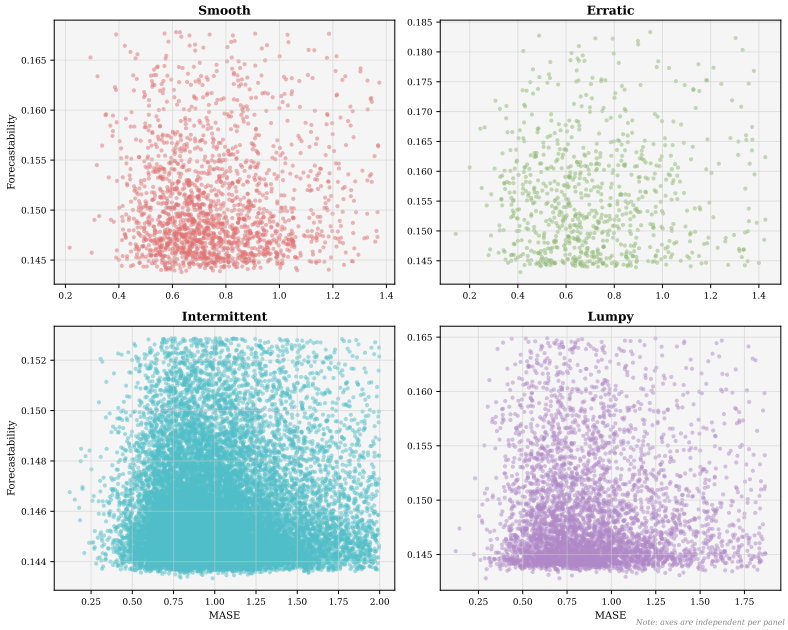


Figure A.3: Scatter plots of forecastability against per-series MASE for the DeepAR-Direct, stratified by demand pattern class.

A.2 Seasonal Strength

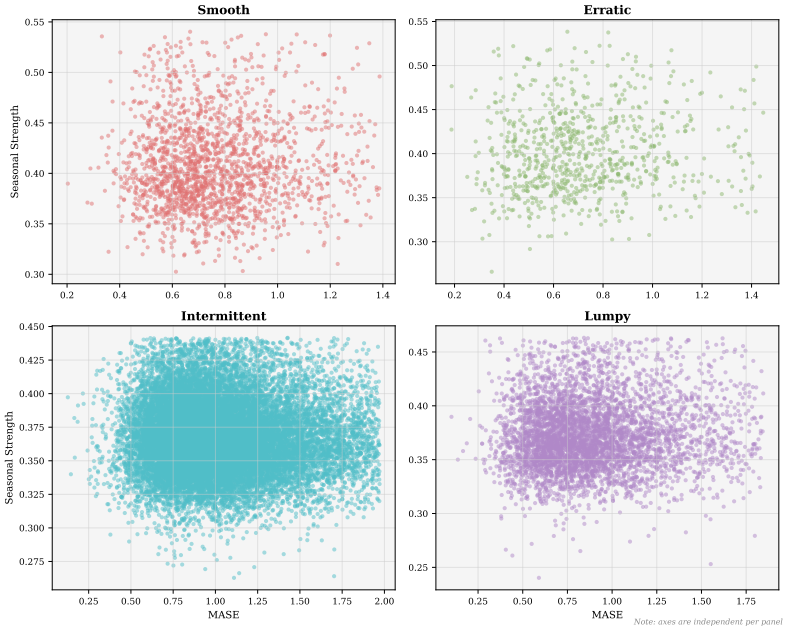


Figure A.4: Scatter plots of seasonal strength against per-series MASE for the LSTM, stratified by demand pattern class.

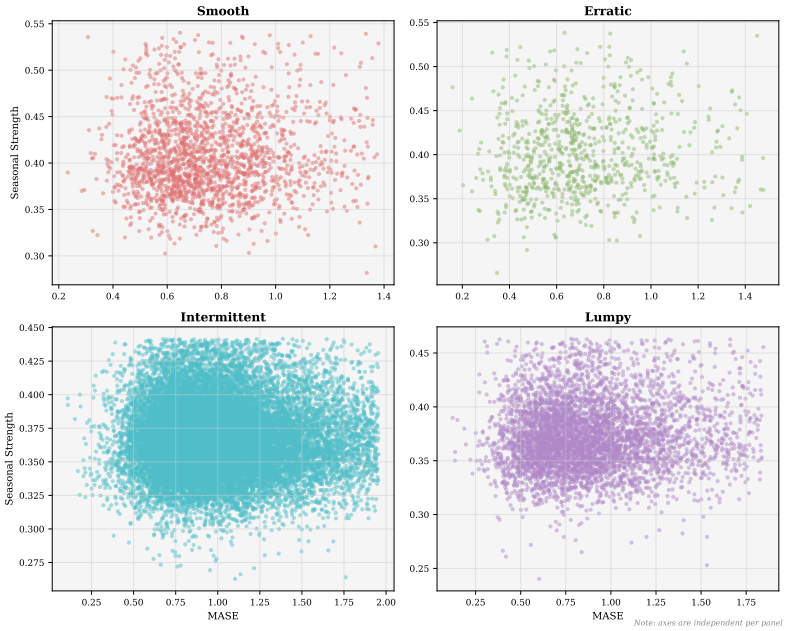


Figure A.5: Scatter plots of seasonal strength against per-series MASE for the TFT, stratified by demand pattern class.

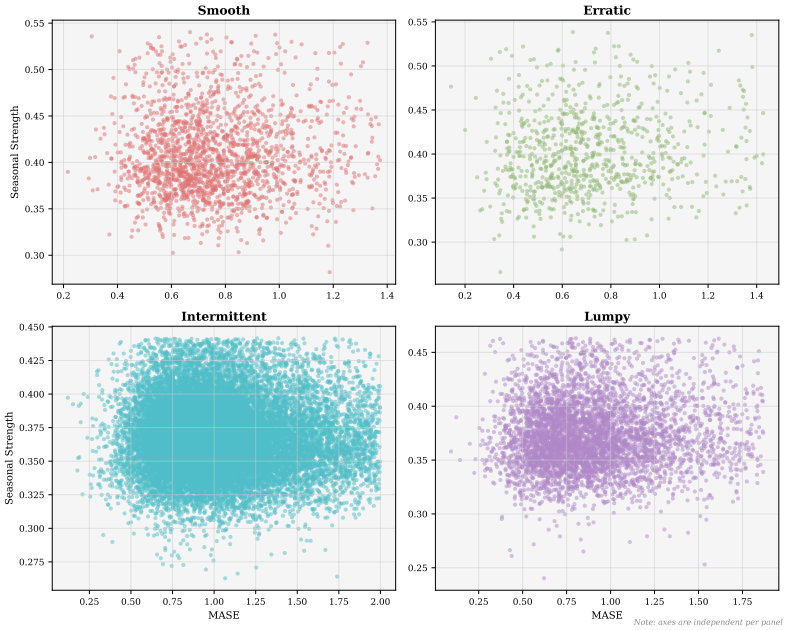


Figure A.6: Scatter plots of seasonal strength against per-series MASE for the DeepAR-Direct, stratified by demand pattern class.

A.3 Autocorrelation Relevance

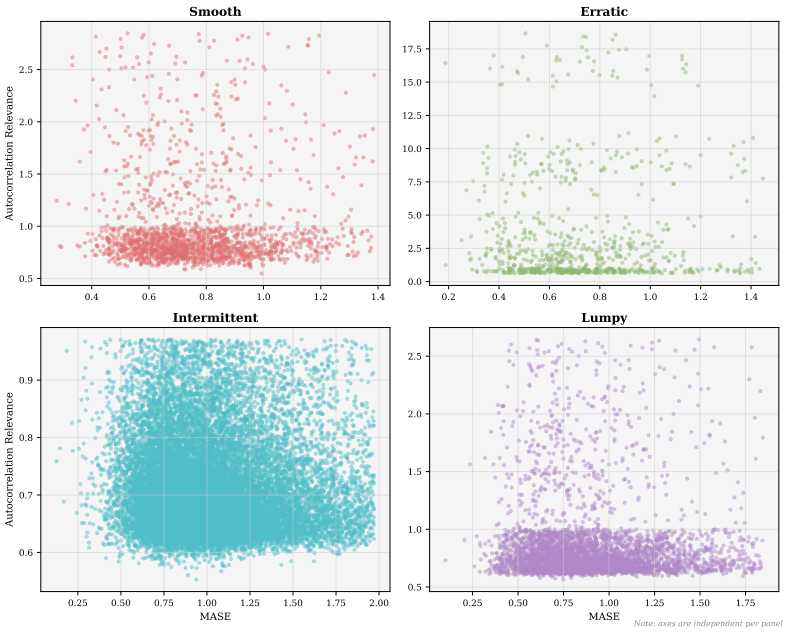


Figure A.7: Scatter plots of autocorrelation relevance against per-series MASE for the LSTM, stratified by demand pattern class.

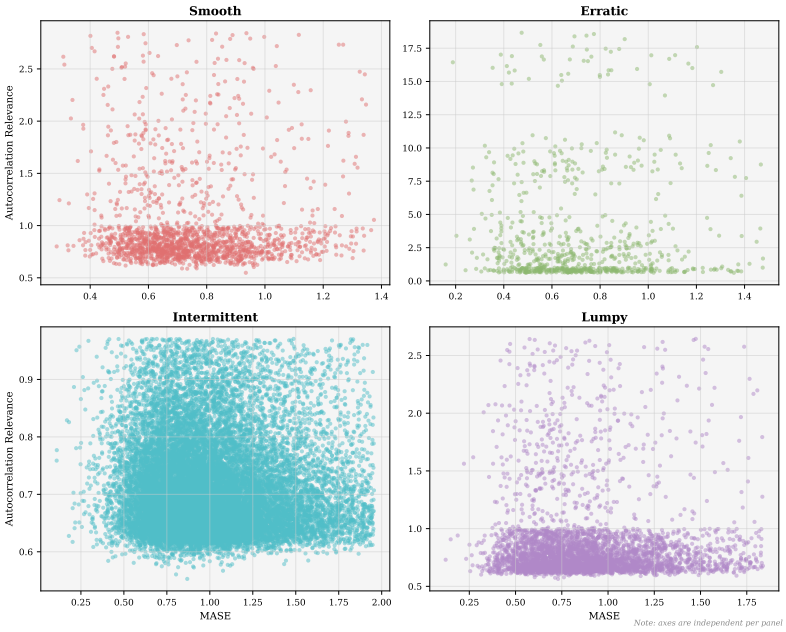


Figure A.8: Scatter plots of autocorrelation relevance against per-series MASE for the TFT, stratified by demand pattern class.

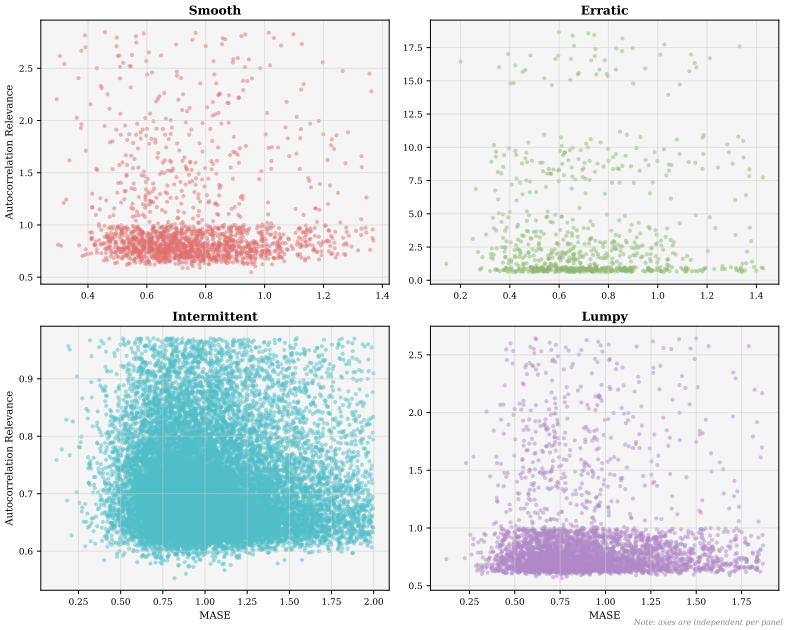


Figure A.9: Scatter plots of autocorrelation relevance against per-series MASE for the DeepAR-Direct, stratified by demand pattern class.

A.4 Complexity

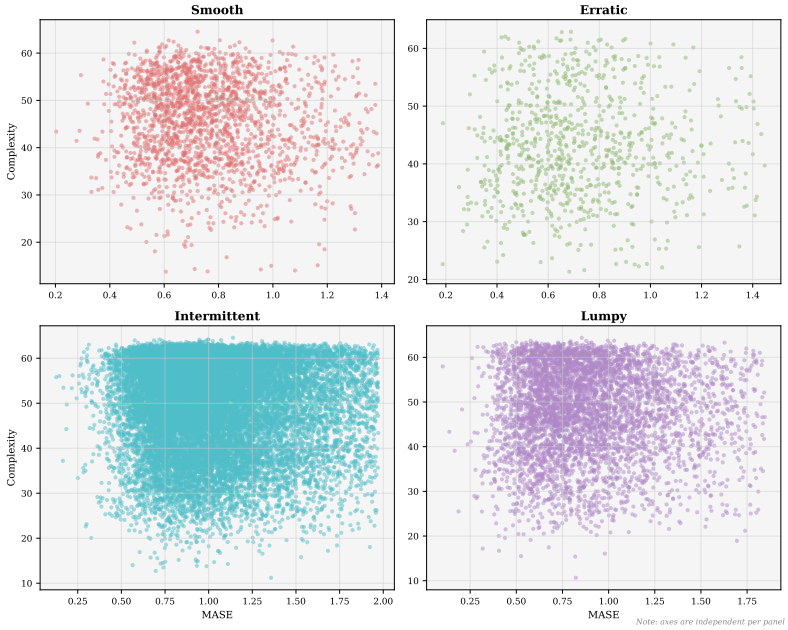


Figure A.10: Scatter plots of complexity against per-series MASE for the LSTM, stratified by demand pattern class.

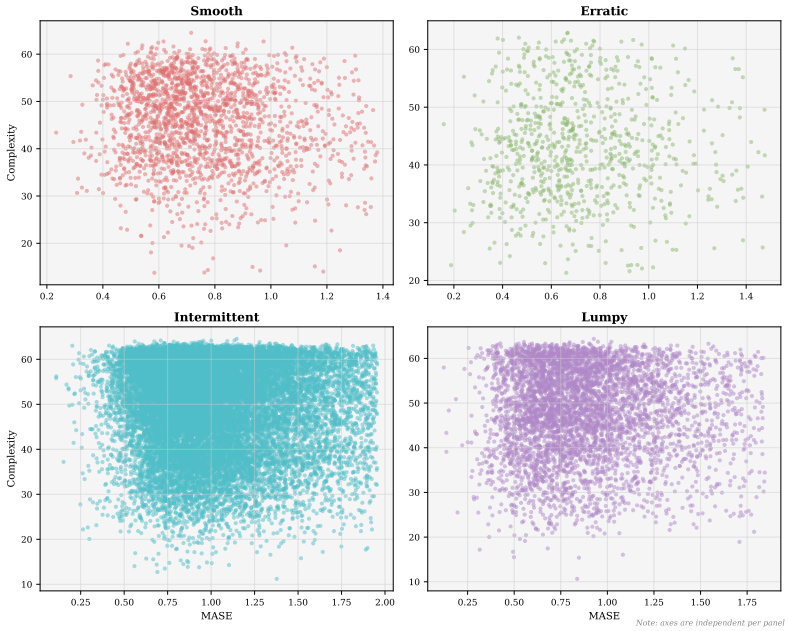


Figure A.11: Scatter plots of complexity against per-series MASE for the TFT, stratified by demand pattern class.

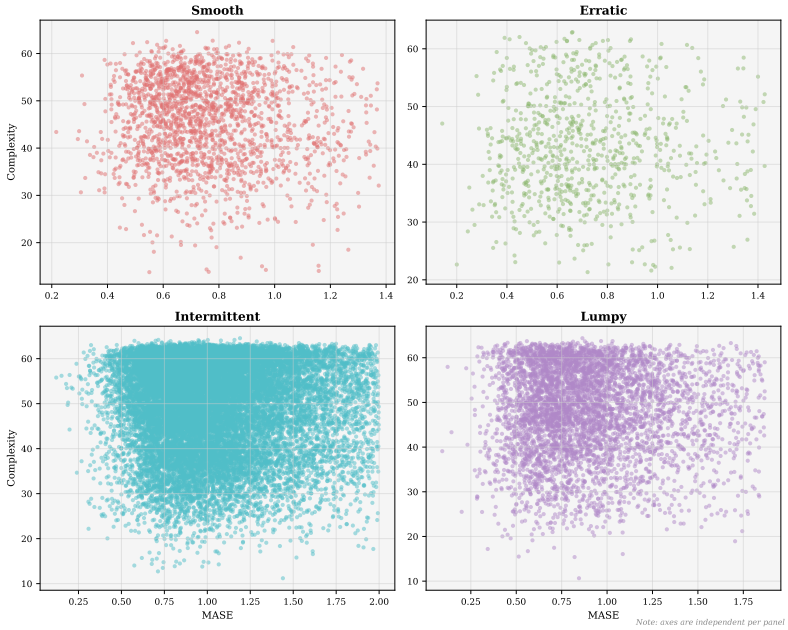


Figure A.12: Scatter plots of complexity against per-series MASE for the DeepAR-Direct, stratified by demand pattern class.