Andy Jin, Jesse Doan, Jasmine Bayrooti **SUNetID's:** andyjin, jdoan21, jasmine7

Abstract

Sales forecasting is important for businesses to gauge their product demand so that they can plan to restock appropriately in advance depending on their estimated future sales. We focus on the dataset provided by Kaggle for the M5 sales forecasting competition. In order to compute accurate predictions for product sales 28 days in advance, we experiment with various statistical techniques and deep learning models to find an optimal approach for this problem. Overall, our results show that LGBM, exponential smoothing, and the N-BEATS model are the most promising approaches for this challenge.

1 Introduction

1.1 Project Motivation

Sales is the driving force behind the success of businesses. It is critical for businesses to have a sense of expected sales so that they can estimate their future demand, ensure they hire a sufficient number of employees, cover operational expenses, and restock supplies in appropriate quantities. Inaccurate forecasts could lead to phenomena such as significant business losses, such as high overhead costs when accumulating a surplus of products in times of low demand or missing profits when there is a shortage of supplies when there is high demand. Thus, it is especially important for large retail stores like Walmart to accurately plan in advance to keep stock of their customers' needs, such as food, toilet paper, etc.

1.2 Problem Statement

In order to address this problem of achieving accurate sales forecasts, we are entering Kaggle's M5 Competition on forecasting the unit sales of Walmart retail goods. The task is to predict the sales volume (i.e. number of goods sold) for each of 3,049 Walmart products, for each of 10 stores, over a validation period of 28 days. The products span 3 categories (Hobbies, Household, Food) and 7 departments across 3 states (CA, TX, WI). Concretely, the input to our algorithm is a sales volume time series for each product over the past 5 years. We use several flavors of a neural network to output the predicted sales volume for each of the next 28 days.

2 Related Work

From a previous iteration of the contest (M4), Redd et al described a state-of-the-art Exponential Smoothing (ES)-RNN model that scored well [10]. The model first employs a pre-processing layer that uses ES, followed by an LSTM layer that updates the parameters of the Holts-Winter model for each time series. Concretely, the first stage of the model performs deseasonalization and adaptive normalization by training seasonality and smoothing parameters for each time series. The second step was an RNN (using LSTM cells) that predicts sales on the deseasonalized data. Finally, the results from the ES and the RNN (stages 1 and 2) are averaged to reduce variance in results. Considering that this approach was successful in the M4 challenge, we hope to achieve good results when applying this technique to the M5 dataset.

Very recent work has shown N-BEATS (Neural Basis Expansion) to also be an effective alternative approach for the M4 competition. In particular, this network utilizes backward and forward residual links and a deep stack of fully-connected layers, which improves the model accuracy by 3% over the past M4 winner. The model does not use time-series-specific components, but rather relies on residual blocks for forecasting [9].

Recent work in audio signalling also indicates the potential of the new WaveNet model, developed by DeepMind for audio synthesis. This model outperformed Google's best text to speech and generates raw wave forms that mimic human sound [8]. Not only is the WaveNet model state-of-the-art for audio generation, but it has also been shown to be promising for similar forecasting tasks like predicting Uber demand in New York City [2] and financial time series forecasting [1].

3 Dataset and Features

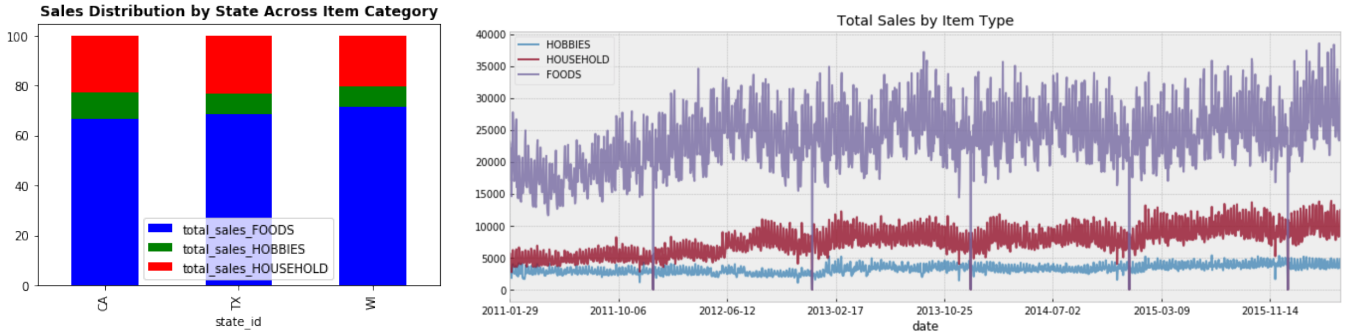
3.1 Datasets

We use the three datasets provided by Kaggle for the M5 sales forecasting competition:

- **sell_prices.csv** $(6,841,121 \times 4)$ — Records the price of the products sold per store and per date.
- **calendar.csv** $(1,969 \times 14)$ — Contains information about the dates on which the products are sold. Includes information for each date, such as event names, event types, day of the week, an ID to identify the year and the week, and whether or not a state provides SNAP benefits on that day.
- **sales_train_validation.csv** $(30,490 \times 1,919)$ — Contains the historical daily unit sales data per product and store for $1,913$ days (`d_1 - d_1913`). **There are 10 unique stores and 3,049 unique items (making up the 30,490 rows).** `d_1 - d_1913` refers to the number of sales for that row item on the nth day for 1913 days. There are three types of items: hobbies, household, and foods.

3.2 Exploratory Data Analysis (EDA)

The sales data come from 10 different stores across the United States: 4 in California, 3 in Texas, and 3 in Wisconsin. The majority of sales came from items labeled foods (68.6%) while the rest came from items labeled hobbies (9.3%) and household (22.0%). Within these item categories, there are 2 hobby-items departments, 2 household-items departments, and 3 food-items departments (with one of the food departments making up 49.3% of the total sales data!). Figure 1(a) (as well as Figure 6(a)) illustrate the distribution of item types across states [7].



(a) Comparison of State to Item Categories

(b) Product Sales Over Time

Figure 1: Analyzing Trends in the Data

Comparing the sales of products from these three categories over time in Figure 1(b) illustrates the prevalence of the food products. The relatively prominent separation between the three items suggests that we could explore different time series models for each category. We concretely consider one food item from the validation set with id `FOODS_3_090_CA_3_validation` and merge its calendar data with real dates to obtain a depiction of its sales over time shown in Figure 6(b) (in the appendix) which reveals that there are often long stretches of days without sales. This influences our project, since we must account for periodic sparsity in the time series. An analysis of the trends for this item's sales with respect to days, weeks, and years using the calendar information is given below:

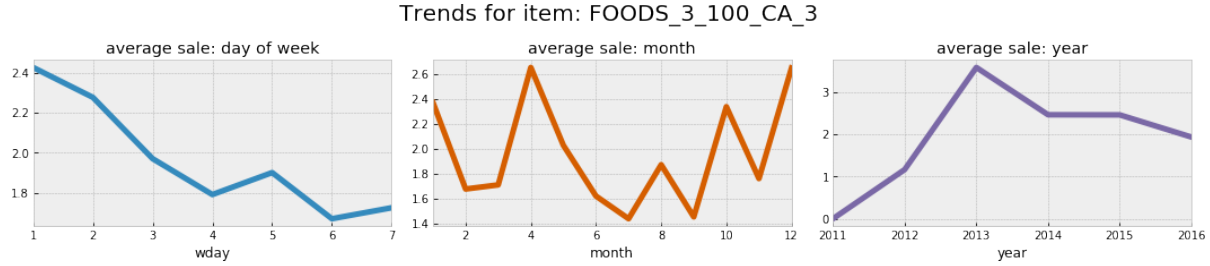


Figure 2: Trends in Item Sales

4 Methodology

4.1 Preprocessing

Before running our models, we performed pre-processing by (1) encoding calendar information as a series with `item ids`, `store ids`, and `event ids`, (2) merging calendar and pricing data, (3) separating the data into train, validation, and

evaluation sets, (4) reshaping to expected submission formatting. Thus, we produced one dataframe encapsulating all of the necessary information from the four provided csv files.

4.2 Baseline Models

We implemented two non-deep learning baseline models: simple average prediction and Light-GBM. For our first baseline, we implemented a model that, for day t, simply predicted the average of the past 30 days sales $(t-30,\ldots,t-1)$. We improved this simple model with an adaptation of a public Light-GBM (LGBM) based kernel that achieved reasonable accuracy [5]. In addition to data pre-processing, we performed feature engineering to extract the following features:

- Generic identifiers such as `item_id, department_id, store_id, sell_price`, etc.
- Demand-related metrics such as the mean, standard deviation, maximum, and minimum demand over varying sliding windows of time (namely, 7, 30, 60, 90, and 180 day frames). Notably, we use these aggregate metrics in lieu of the raw sales across the $1,913$ training time steps to combat the aforementioned sparsity problem.
- Price-related metrics, such as shifts in price, mean, standard deviation, maximum, and minimum prices rolling over the same time windows
- Time-related features such as the day of the week, whether or not it is the weekend, month, etc.

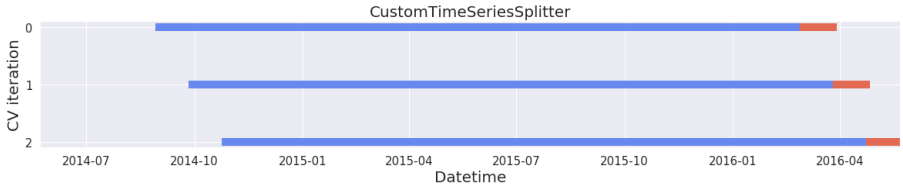


Figure 3: Three Folds for Validation

In total we generated 21,434,470 training samples and 853,720 testing samples for the LGBM model. We then trained on 1.5-years worth of data (547 days) and validated on the following 28 days with three folds as illustrated in Figure 3, where the blue bars correspond to the training period and orange bars correspond to validated periods. The validation periods always come after the training.

4.3 LSTM Model

Long Short Term Memory (LSTM) models are kinds of recurrent neural networks that tend to perform well on sequential and time series data. We assembled a simple encoder/decoder model (LSTM architecture in appendix). We used a pretrained Keras callback model to save model checkpoints each time the model achieves greater performance, as measured by our RMSSE metric, and log incremental tensorboard outputs. We trained the model with a batch size of 64, learning rate of 10^{-3}, Adam optimizer, hidden layer size of 32, and dropout probability of 0.2 for 25 epochs. We used a simple 20/80% validation-training split before fine tuning the model.

4.4 WaveNet Model

WaveNet is another powerful approach, which combines convolutional layers with audio signal processing methods and has been shown to be promising for time series tasks, as discussed in the Related Work section. Like the LSTM model, we used a pretrained WaveNet Keras callback model to save model checkpoints each time the model improves on the validation set while logging incremental tensorboard outputs [3]. We trained this model with a batch size of 256, learning rate of 3e-4 and Adam optimizer. For the convolutional layers of the model, we used 32 2x2 filters with causal padding as well as dilation rates of $0, 2, 4, 8, ..., 2^8$ for each layer respectively. We used dropout probability of 0.2 and trained for 25 epochs. Again, we used a simple 20/80% validation-training split before fine tuning the model while extracting the last 28 time steps as the training target. The full architecture of the WaveNet model is provided in the Appendix. It consists of a series of Conv1D layers, followed by a Dense, Dropout, Dense, and Lambda layer.

4.5 N-BEATS Model

Neural Basis Expansion Analysis for interpretable Time Series forecasting (N-BEATS) is a novel deep interpretable architecture which relies on residual blocks, a type of layer that not only feeds into the next layer but also directly into layers that are some number of layers away from the initial layer. For M5, we adapted the original paper's model

architecture, which focuses on doubly residual stacks and is shown in the appendix [4]. In this architecture, there are multiple stacks which lead to the model output or the global forecast. Within each stack, there is a stack of blocks that contribute to the stack output or stack forecast. Furthermore, within each block, each of them contain several fully-connected layers which outputs basis expansion coefficients for backward (denoted backcast) and forward (denoted forecast). The backcasts and forecasts of each block feed into the subsequent blocks and the overall stack forecast. For our implementation, we use an Adam optimizer with a learning rate of 10^{-3}, hidden layer of size 1024 and theta dimensions of 8 x 8.

4.6 Exponential Smoothing

We experimented with triple Exponential Smoothing (ES) or Holt-Winters to smooth the demand values for a given product as a preprocessing step for our best performing prediction model, the LGBM baseline. As mentioned in the Related Work section, ES has been shown to enhance performance as it captures the idea that the forecast of a next step is approximately the forecast of the previous step adjusted by part of the previous error. Naturally, this has a smoothing effect and its combination with the LSTM model for the M4 sales forecasting competition led to significant improvements in forecasting accuracy.

Specifically, for a given series in the data frame, we aim to predict two future points while taking seasonality into account. We break this down into the intercept l and slope m, which we aim to predict using the assumption that the future direction of changes in the time series depends on previous weighted changes using the following formulas [6]:

$$l_t = \alpha * (y_t - s_{t-L}) + (1 - \alpha) * (l_{t-1} + b_{t-1}) \qquad s_t = \gamma * (y_t - l_t) + (1 - \gamma) * s_{t-L}$$

$$b_t = \beta * (l_t - l_{t-1}) + (1 + \beta) * b_{t-1} \qquad \hat{y}_{t+m} = l_t + m * b_t + s_{t-L+1+(m-1)modL}$$

We apply this smoothing to the series for FOODS_3_634_WI_2_validation and pass this through the LGBM model.

5 Results and Discussion

The following table describes our models' performances using the root mean squared scaled error (RMSSE), which is Kaggle's metric for results evaluation for this competition, and is chosen primarily because it is scale-independent and optimizes for the mean (since we are predicting average demand).

Model	Approx. Kaggle Ranking	Test Set (Kaggle) RMSSE
Past 30-Day Average	25th Percentile	1.071
Light GBM	60th Percentile	0.697
RNN-LSTM	8th Percentile	1.761
WaveNet	8th Percentile	1.662
N-BEATS	22th Percentile	1.164

Overall, these results were not quite as we expected. Due to their greater complexity and representation power, we had anticipated that the LSTM and WaveNet models would outperform our simpler baseline models on the test set, but they actually achieved a worse score when submitted on Kaggle. Nevertheless, these models did achieve significantly lower training errors (0.479 for RNN-LSTM and 0.483 for WaveNet), which suggests that they have greater representation power but may be overfitting to the training set.

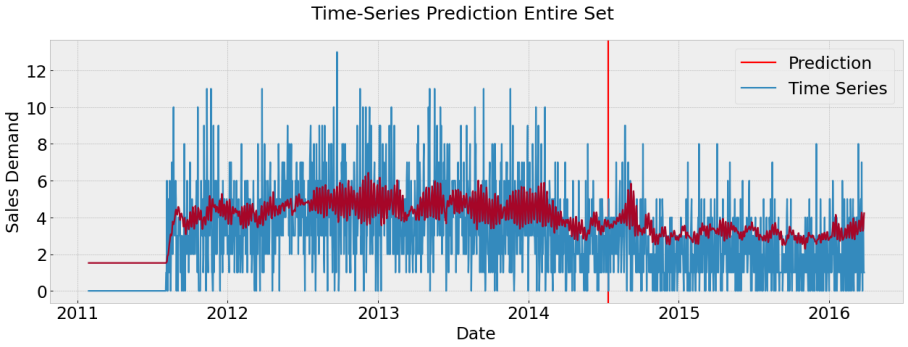


Figure 4: LSTM Prediction Over Entire Series

We investigated the LSTM's performance further and found that the model seemed to be generating decent predictions over the entire training set. This is illustrated by the prediction curve closely following the actual time series values with

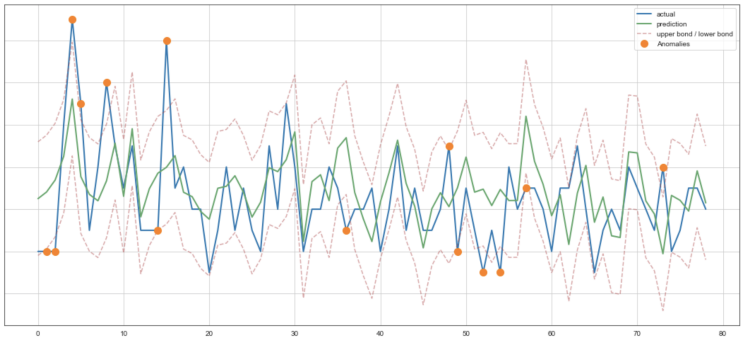
some noise as shown in Figure 4. This confirms that our model is correctly learning and generating decent predictions, though it may struggle with extreme demand values. We anticipate achieving better results with this LSTM model after tuning the learning rate, hidden size, and other hyperparameters. Furthermore, considering the high performance of the LGBM model, we hypothesize that perhaps passing in the features for the LGBM model could enhance the LSTM's performance.

For the WaveNet model, we attribute its high RMSSE score to the difference between the lower dimensional M5 training dataset and the more unstructured and high dimensional audio signals. WaveNet is typically useful for different kinds of data and, despite its successes for predicting Uber demand and financial time series forecasting, it is likely that this architecture is not very well suited to our task.

The N-BEATS model performed the best of the deep learning models we tried. As described in the Related Work and Methodology sections, N-BEATS achieved 3% greater accuracy than the ES-RNN model which won the M4 challenge. Thus, it is not surprising that this method performed better than the LSTM and WaveNet models.

Finally, we achieved good results with the triple ES approach on a given product series as shown in Figure 5 (a) as the model values nicely smooth over the noisy time series data. Judging by this graph, the triple ES model is able to successfully approximate the initial time series, capture daily seasonality, the general downward trend and even some anomalies. Clearly the model reacts to many changes in the series structure but then tends to return to the normal values and essentially "forget" the past. Thus, we concluded that this technique is promising as it enables anomaly detection and smooths the data, so we decided to apply it to a series through our best model, the LGBM predictor. As shown in Figure 5 (b), the predicted demand matches the actual demand fairly closely.

(a) Triple ES Applied to Product Series



(b) ES+LGBM Applied to Product Series

Figure 5: Using ES over a product series and incorporating into the LGBM model

6 Conclusion and Future Work

As shown in the Results and Discussion section, after experimenting with simple averaging, LGBMs, feature engineering, RNN-LSTMs, WaveNet and N-BEATS models, and exponential smoothing, we observe that the LGBM, exponential smoothing, and N-BEATS are the most effective methods for predicting demand for products 28 days in advance. The LGBM approach achieved the lowest RMSSE score of 0.697 and our results from the ES+LGBM model suggest that such a combination would also perform quite well. Of the deep learning models, N-BEATS performed the best, likely because of its specialization for time series data and focus on residual blocks to allow for series outputs to feed back in to higher layers. Furthermore, we expect that the LSTM model would also generate good predictions (when properly tuned) because of its long-term "memory" properties that serve a similar purpose.

For future work, we can construct a separate model for each time series at the location or product category aggregation level, training parameters separately. We would also like to apply the ES+LGBM approach to the entire training set to see if we achieve better results when smoothing the demand values. If so, we would like to apply exponential smoothing to our other models. We would also like to further investigate the relatively poor performance from the LSTM model and try tuning the hyper-parameters and passing in the features used for the LGBM model. Additionally, we hope to try an encoder-decoder approach for automatic feature engineering, which Wu et al deems promising for time series [11].

Overall, this project significantly improved our understanding of sequence models and working with time series data and we are looking forward to exploring future possibilities with this work!

7 Contributions

All three authors contributed equally to the project. Jesse worked on the code infrastructure and running the models, Andy worked on LGBM feature generation and the N-BEATS model, and Jasmine worked on the exploratory data analysis and exponential smoothing. All three authors contributed to model architecture development and tuning, result analysis, and writing the reports.

8 Acknowledgements

We would like to thank Jo Chuang for his helpful suggestions and feedback during our meetings and mentorship throughout this project. We are also grateful to the entire CS 230 teaching staff for a very fun and interesting quarter delving into deep learning.

References

[1] Anastasia Borovykh, Sander Bohte, and Cornelis W. Oosterlee. "Conditional time series forecasting with convolutional neural networks". In: (2018). URL: https://arxiv.org/pdf/1703.04691.pdf.

[2] Long Chen, Konstantinos Ampountolas, and Piyushimita (Vonu) Thakuriah. "Predicting Uber Demand in NYC with Wavenet". In: (2020). URL: http://eprints.gla.ac.uk/199034/.

[3] James Colless. ***M5 - RNN/Wavenet/N-BEATS Approach***. URL: https://www.kaggle.com/jcolless/m5-rnn-wavenet-n-beats-approach.

[4] Fabien. ***N-BEATS Basics***. URL: https://www.kaggle.com/mpware/n-beats-basics.

[5] harupy. ***M5 Baseline***. URL: https://www.kaggle.com/harupy/m5-baseline.

[6] Rodrigo de Lima Oliveira. ***Time Series Analysis - ES+SARIMAX+XGB+LGBM***. URL: https://www.kaggle.com/rodrigolima82/time-series-analysis-es-sarimax-xgb-lgbm.

[7] Rob Mulla. ***M5 Forecasting - Starter Data Exploration***. URL: https://www.kaggle.com/robikscube/m5-forecasting-starter-data-exploration.

[8] Aäron van den Oord et al. "WaveNet: A Generative Model for Raw Audio". In: ***CoRR*** abs/1609.03499 (2016). arXiv: 1609.03499. URL: http://arxiv.org/abs/1609.03499.

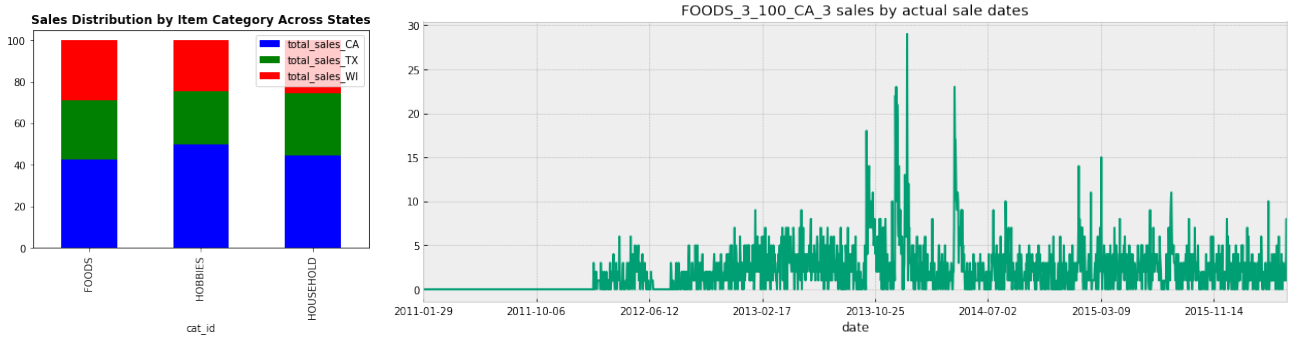
[9] Boris N. Oreshkin et al. "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting". In: (2020). URL: https://openreview.net/forum?id=r1ecqn4YwB.

[10] Andrew Redd, Kaung Khin, and Aldo Marini. "Fast ES-RNN: A GPU Implementation of the ES-RNN Algorithm". In: ***CoRR*** abs/1907.03329 (2019). arXiv: 1907.03329. URL: http://arxiv.org/abs/1907.03329.

[11] Neo Wu et al. ***Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case***. 2020. arXiv: 2001.08317 [cs.LG].

9 Appendix

9.1 Supplementary Exploratory Data Analysis Figures



(a) Comparison of Item Category to State

(b) Sales for a Given Product Over Time

Figure 6: More Analysis of Trends in the Data

9.2 Model Architectures

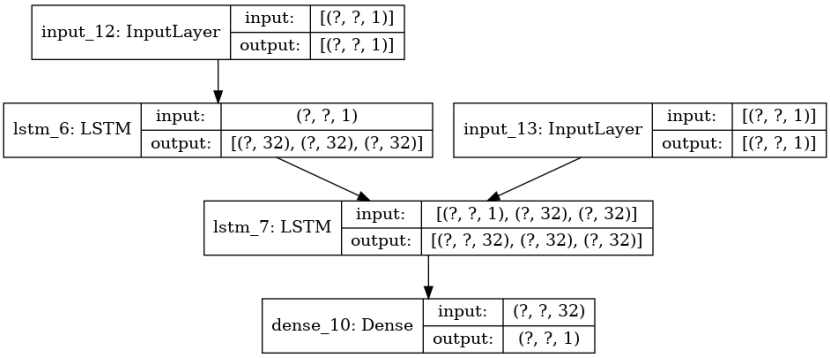


Figure 7: LSTM Model Architecture

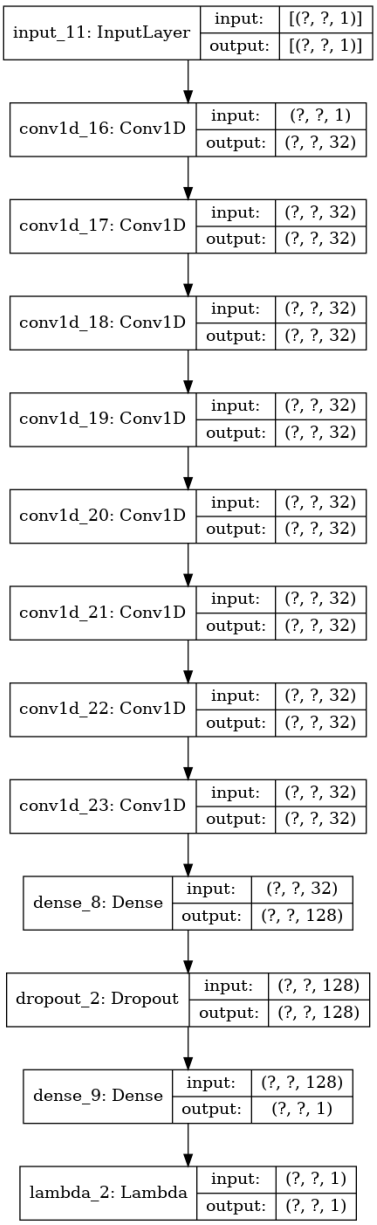


Figure 8: WaveNet Model Architecture

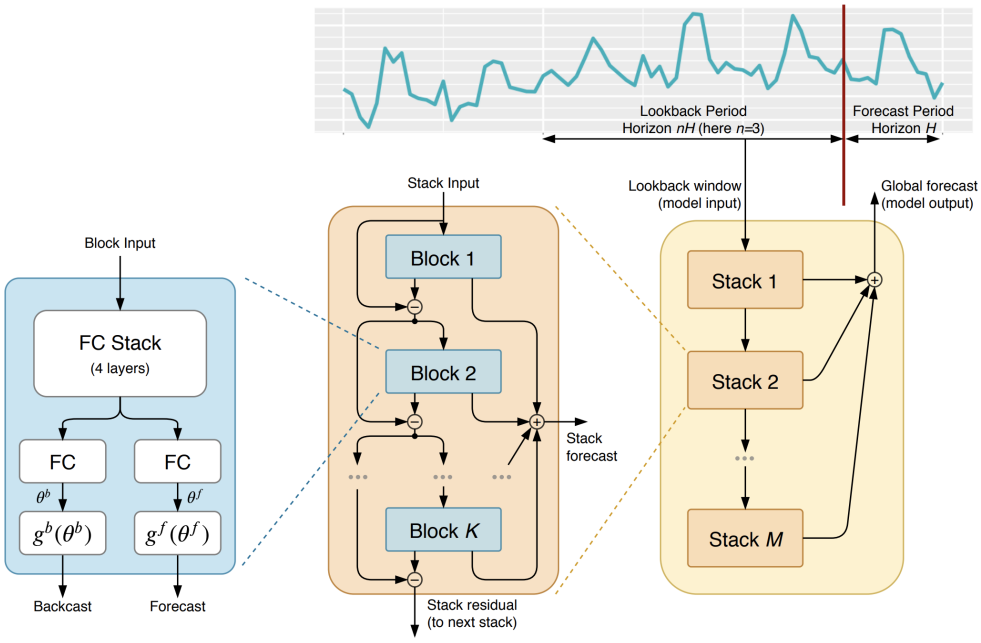


Figure 9: N-BEATS Model Architecture