Forecasting Market Prices using DL with Data Augmentation and Meta-learning: ARIMA still wins!

Vedant Shah
APPCAIR
BITS Pilani, K K Birla Goa Campus
f20180566@goa.bits-pilani.ac.in

Gautam Shroff
TCS Research
New Delhi
gautam.shroff@tcs.com

Abstract

Deep-learning techniques have been successfully used for time-series forecasting and have often shown superior performance on many standard benchmark datasets as compared to traditional techniques. Here we present a comprehensive and comparative study of performance of deep-learning techniques for forecasting prices in financial markets. We benchmark state-of-the-art deep-learning baselines, such as NBeats, etc., on data from currency as well as stock markets. We also generate synthetic data using a fuzzy-logic based model of demand driven by technical rules such as moving averages, which are often used by traders. We benchmark the baseline techniques on this synthetic data as well as use it for data augmentation. We also apply gradient-based meta-learning to account for non-stationarity of financial time-series. Our extensive experiments notwithstanding, the surprising result is that the standard ARIMA models outperforms deep-learning even using data augmentation or meta-learning. We conclude by speculating as to why this might be the case.

1 Introduction

Forecasting prices in financial markets has long been attempted using a variety of statistical as well as modern machine-learning techniques; in theory markets are ‘efficient’, and should not be predictable, making this task potentially ill-posed to start with. Nevertheless, financial markets are rife with algorithmic trading engines that aim at precisely this, attempting to discover any lingering inefficiencies that could yield a usable trading signal.

The behaviour of stock markets is a result of complex combination of several trading heuristics followed by a large number of groups of traders ranging from big financial firms to individual investors, each with varying impact. Previous works (Wang (2015b,a)) have shown that these trading strategies and the behaviour arising from their combination can be modeled as a set of mathematical equations for the excess demand function using fuzzy logic. These set of equations are capable of modelling the real-world financial markets to a good extent as shown in Wang (2015c). These models can then be used to generate data of desired amount which will follow the behaviour arising from the combination of all the trading heuristics considered while building the model.

Present day deep learning models require large amounts of data to train. The foremost problem in using deep-learning for market data forecasting is the lack of availability of large datasets; after all, there is only so much historical data available. We hypothesize that synthetic data can be used to improve the performance of existing Deep Learning models on forecasting real-world market data. This is based on the premise that training on even very dissimilar data can help in improving the performance of deep learning models as shown in Malhotra et al. (2017).

I (Still) Can’t Believe It’s Not Better Workshop at NeurIPS 2021.

Meta-learning is a machine learning paradigm wherein a model is trained on multiple similar tasks; for example, tasks having slightly different data distributions. Gradient-based meta-learning using MAML introduced in Finn et al. (2017) is a model-agnostic procedure wherein a base model is adapted using the training data of each task via a few gradient steps. During meta-training, the based model is trained in an outer-loop, again using gradient descent, so that its post-adaptation performance on the test data of the meta-training tasks is optimized. The goal is to initialize the model with a set of parameters which can easily be adapted to different downstream tasks, potentially helping the model to generalize to out of distribution data. Financial data can be thought of as undergoing frequent changes of distribution as economic conditions and investor sentiments fluctuate due to global and national events

We present a comprehensive study in which we attempt using meta-learning and synthetically generated data for data augmentation, to improve performance of deep-learning models for forecasting financial market data. We report results across various deep learning models: MLPs, LSTMs, Transformers (Vaswani et al. (2017)) and two state of the art time-series forecasting models: NBEATS (Oreshkin et al. (2020)) and Temporal Fusion Transformers (TFT) (Lim et al. (2019)). Surprisingly, we find that none of these models is better than the traditional ARIMA model, even using data augmentation or applying meta-learning.

2 Related work

Makridakis and Hibon (2000) and Makridakis et al. (2018) provide an in depth review of the performance of various statistical and Machine Learning algorithms for time-series forecasting in the M3 competition. Recently, Oreshkin et al. (2020) proposed N-BEATS, a deep learning univariate time-series forecasting model which is capable of beating statistical approaches on the M4 benchmark. Lim et al. (2019) introduced Temporal Fusion Transformer, a state of the art multivariate time-series forecasting model which shows improved performance as compared to ARIMA and ETS on a number of real-world tasks. DeepAR proposed in Flunkert et al. (2017) uses auto-regressive RNN for probabilistic forecasting. Oreshkin et al. (2021) studies zero-shot learning for univariate time series forecasting using Model Agnostic Meta Learning as proposed in Finn et al. (2017) on N-BEATS. Another very recent work, Elsayed et al. (2021) shows that traditional forecasting approaches such as Gradient Tree Boosting Models are still capable of beating a number of state of the art deep learning approached on an array of different datasets.

3 Methodology

Wang (2015b) uses fuzzy logic to develop a mathematical model of the excess demand of the stock market by taking different trading heuristics into account. This mathematical is then used to generate synthetic data using:

$$\ln(p_{t+1}) = \ln(p_t) + \sum_{i=0}^{M} a_i(t) ed_i(x_t) \tag{1}$$

where p_t denotes the price of a share at a give time-step, $a_i(t)$ denotes the influence of a particular heuristic, and $ed_i(x_t)$ denotes the excess-demand function for the heuristic with x_t being a time dependent variable. We consider synthetic data generated from **Heuristic 1** discussed in Wang (2015b) for all purposes. Using fuzzy logic, excess demand can be modeled into the expression given below; twelve such heuristics are discussed in Wang (2015b).

$$ed_1(x_{1,t}^{(m,n)}) = \frac{\Sigma_{i=1}^{7} c_i \mu_{A_i}(x_{1,t}^{(m,n)})}{\Sigma_{i=1}^{7} \mu_{A_i}(x_{1,t}^{(m,n)})} \tag{2}$$

where $x_{1,t}^{(m,n)}$ is the ratio of the natural logarithm of the two moving averages of length m and n of the stock price. The price of a stock at time step t is denoted by p_t. μ_{A_i} are a fuzzy membership functions for each of the 7 rules as in Wang (2015b), describing when each rule applies. The moving average of length n and the ratio of the natural logarithms of two moving averages of length m and n can then be calculated as: $\bar{p}_{t,n} = \frac{1}{n}\sum_{i=0}^{n-1} p_{t-1}$ and $x_{1,t}^{(m,n)} = \ln(\frac{\bar{p}_{t,m}}{\bar{p}_{t,n}})$

We experiment with two different techniques in an attempt to improve market data forecasting using synthetic data:

Data Augmentation: We augment the real world data with an equal amount of synthetic data during training the models. For each epoch, batches of real-world data and synthetic data are fed randomly to the deep learning model until all the batches of both synthetic and real world data are exhausted. We use a lookback length of 20 for the real-world data and a lookback length of 5 for the synthetic data. For models with fixed input length such as MLP and NBEATS, we interpolate the lookback windows of the synthetic data to make their length equal to those of the real-world data.

Algorithm 1 Data Augmentation

```
Initialize the synthetic dataloader $\mathcal{S}$, the real-world dataloader $\mathcal{R}$ with appropriate batch sizes
    and the number of epochs $e$
$n \leftarrow len(\mathcal{S})$; $m \leftarrow len(\mathcal{R})$
for $i$ in $e$ do
    $s \leftarrow 0$; $r \leftarrow 0$
    while $s < n$ or $r < m$ do
        $a \sim U(0, 1)$
        if $a > 0.5$ then
            if $s < n$ then
                $batch \sim \mathcal{S}$
                Input the lookback window into the model
                Calculate the loss on the batch
                Backpropagate on the loss
                $s \leftarrow s + 1$
            end if
        else
            if $r < m$ then
                $batch \sim \mathcal{R}$
                Input the lookback window into the model
                Calculate the loss on the batch
                Backpropagate on the loss
                $r \leftarrow r + 1$
            end if
        end if
    end while
end for
```

Meta Learning: We create the tasks for meta-training from the training part of a dataset and those for meta-testing from testing part of the dataset. For synthetic data, these divisions are made within each of the 40 time-series. For creating each task for meta-training, a time-step d is chosen randomly from the training data. Let l be the total length of the window ($lookback + prediction$). For each such time-step d, k such windows, $[d - l - j, d - j]$ are sampled randomly from before d, where $j \sim U[1, a]$. This forms the training set of the task. Similarly, r windows are sampled randomly after d which forms the test set of the task. The same procedure is used on the testing dataset to create meta-testing tasks. For synthetic data, meta-training and meta-testing tasks are created from each series and then merged. All the deep learning models are then trained using meta-learning on both real-world and synthetic data individually using First Order Model Agnostic Meta Learning algorithm (Nichol et al. (2018)).

4 Empirical evaluation

4.1 Materials

We use run experiments on three datasets: one **synthetic** and two **real world** datasets.

Synthetic Data: We use equations (1) and (2) with $(m, n) = (1, 5)$ to generate 40 synthetic time series each containing 500 time-steps. Each of these time-series is initialized with different seeds and a random-walk for 100 time-steps. For normal training and testing, 36 time-series are used as

Table 1: Results on three datasets **SN** - Synthetic, **FR** - Forex, and **BN** - Banknifty for **NTR** - Normal Training, **DA** - Data Augmented Training and **M2L** - Meta Learning. **OS** - One step prediction, **TS** - Tens step prediction, **Average DL** - Average of performance of all five deep learning models

			ARIMA		Average DL		Best DL Model		Best DL
			RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	Model Name
SN	NTR	OS	0.5496	0.0387	0.3837	0.0127	**0.2313**	**0.0102**	NBEATS
		TS	0.8561	0.0527	0.8126	0.0454	**0.7591**	**0.0409**	NBEATS
	M2L	OS	-	-	0.3655	0.0262	0.2961	0.0219	NBEATS
		TS	-	-	0.7510	0.0530	0.7205	0.0514	LSTM
FR	NTR	OS	**0.1982**	**0.0027**	0.2962	0.0029	0.2734	0.0027	LSTM
		TS	**0.4754**	**0.0056**	0.5923	0.0058	0.5648	0.0055	LSTM
	DA	OS	-	-	0.2849	0.0028	0.2609	0.0027	Transformer
		TS	-	-	0.5980	0.0059	0.5680	0.0055	LSTM
	M2L	OS	-	-	0.3443	0.0038	0.3235	0.0036	MLP
		TS	-	-	0.6579	0.0069	0.6029	0.0064	LSTM
BN	NTR	OS	**435.18**	**0.0131**	816.63	0.0187	704.68	0.0158	NBEATS
		TS	**1196.09**	**0.0314**	1491.46	0.0337	1396.34	0.0312	LSTM
	DA	OS	-	-	858.85	0.0200	728.65	0.0165	MLP
		TS	-	-	1486.22	0.0339	1397.29	0.0313	LSTM
	M2L	OS	-	-	877.94	0.0238	784.62	0.0213	MLP
		TS	-	-	1674.17	0.0434	1562.72	0.0403	LSTM

training data and 2 time-series are used for validation and testing data each. For meta learning, the first 400 time-steps of each time series are used as the meta-training data and the last 100 time-steps are used as the meta-testing data.

Banknifty Data: Banknifty data is a time-series of closing values of the banknifty index, recorded every week day from 1st January, 2016 to 31st May, 2021. After cleaning the data we split the data into a training time series containing 1050 time-steps and testing and validation time series, each containing 144 time-steps. For meta learning the the time series consisting of 1050 time-steps is used to create tasks for meta-training and the rest of the time series consisting of 288 time-steps (144 + 144) is used for creating meta-testing tests.

Forex Data: Forex Data is a time-series of spot prices of 1 INR in USD, recorded every weekday from 1st January, 2010 to 14th August, 2020. The total length of the time series is 2562 time-steps of which the the first 2050 time-steps are used as the training data, time steps 2045 to 2306 are used as the validation data and time-steps 2301 to 2562 are used as testing data. For meta learning, the first 2050 time-steps are used for creating meta-training tasks and the remaining data is used to create meta-testing tasks.

4.2 Results

Table 1 compares[1] the performance of deep learning models trained using different training techniques with ARIMA. We use a multitude of deep learning models ranging from simple models to current state of the art models for time-series forecasting. Results were obtained for: MLP, LSTM, Transformers , NBEATS and TFT. All the experiments were performed using a 12GB NVIDIA Tesla K80 GPU, 12GB RAM and an Intel(R) Xeon(R) CPU with 2 cores. The results have been reported for two well known metrics: **Root Mean Squared Error (RMSE)** and **Mean Absolute Percentage Error (MAPE)**.

The lookback window length for the synthetic dataset has been taken as 5 since each time-step depends on the last 5 time-steps and the lookback window length for both the real-world datasets

[1] Transformer suffers from exploding gradients when Meta Learning. TFT suffers from exploding gradients when Meta Learning on Banknifty Data. These cases aren't considered while calculating average performances.

Table 2: Rollout Testing with and without Data Augmentation for non-autoregressive models

			MLP		NBEATS		TFT	
			RMSE	MAPE	RMSE	MAPE	RMSE	MAPE
SN		OS	0.2754	0.0127	0.2315	0.0102	0.4361	0.0225
		TS	0.9691	0.0522	0.9957	0.0529	0.8567	0.0479
FR	NTR	OS	0.2935	0.0029	0.3233	0.0033	0.3936	0.0038
		TS	0.6030	0.0059	0.7699	0.0079	0.7036	0.0068
	DA	OS	0.3093	0.0030	0.2984	0.0029	0.3005	0.0030
		TS	0.6877	0.0064	0.5982	0.0058	0.6784	0.0067
BN	NTR	OS	730.23	0.0164	707.82	0.0159	977.06	0.0233
		TS	1480.58	0.0309	1413.56	0.0318	1431.35	0.0339
	DA	OS	727.78	0.0165	851.85	0.0209	1006.77	0.0239
		TS	1462.83	0.0306	1551.39	0.0374	1583.22	0.0371

has been taken as 20 which was arrived at by experimenting with different lookback lengths. The lookback windows are locally normalized before passing through the model which allows the real-world and the synthetic data to be combined during data augmentation. We evaluate each model for a prediction horizon of 10 time-steps and the results are calculated after de-normalizing the predictions. We further decompose the results obtained to get results for one-step ahead prediction, five step ahead prediction and 6th to 10th step prediction. Results for one step (OS) and ten step (TS) ahead predictions have been reported.

4.2.1 Rollout Testing

We further experiment with try rollout testing for models trained with and without data augmentation on non-autoregressive models (MLP, NBEATS and TFT). In rollout testing, we take into account the previous n points to predict the next time-step where n is the length of the lookback window. The lookback window is of fixed length and keeps sliding one step at a time as and when new predictions are made. We take n = 5 for synthetic data in an attempt to make it easier for the predictor to capture the actual relationship between the time-steps (remember that we use n = 5 in equation 2 for generating the data). $n = 20$ for Banknifty and Forex datasets. The results are presented in Table 2.

Clearly, state of the art deep learning models for time-series forecasting fail to beat classical approaches such as ARIMA on the real-world data, even after data augmentation and meta learning. Deep Learning models successfully beat ARIMA on synthetic data which is deterministic. Real-world financial data is very noisy and chaotic. Machine Learning models rely on a global feedback signal or a recent signal for optimization. Due to the (financial) data being noisy and chaotic and the fact that deep learning models are often trained on multiple time-series, the feedback signals are also very noisy leading to poor optimization. The synthetic data on the other hand is deterministic, hence leading to noiseless signals due to which deep learning models perform better on synthetic data. ARIMA on the other hand relies on a purely local signal computed every time it's applied. A local signal is much stronger when there is more noise in the longer signals, e.g., because the inefficiencies that lead to potential trading signals are *not* similar over time and change continuously.

5 Conclusion

We present a comprehensive and comparative study on performance of different Deep Learning techniques and ARIMA, a classical approach on market data forecasting. While current state of the art deep models are often capable of outperforming classical forecasting techniques, they fail to do so on financial data - an initally surprising negative result. We attribute the reason to the fact that financial data is inherently chaotic. Autoregressive models such as ARIMA in general perform better for prediction when there is less reason to believe that the regularities enabling prediction are themselves non-stationary and cannot therefore be exploited by machine-learning models. We find that this is indeed true for the financial data that we experimented on.

References

Elsayed, S., Thyssens, D., Rashed, A., Schmidt-Thieme, L., and Jomaa, H. (2021). Do we really need deep learning models for time series forecasting? *ArXiv*, abs/2101.02118.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

Flunkert, V., Salinas, D., and Gasthaus, J. (2017). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *ArXiv*, abs/1704.04110.

Lim, B., Arik, S. Ö., Loeff, N., and Pfister, T. (2019). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *ArXiv*, abs/1912.09363.

Makridakis, S. and Hibon, M. (2000). The m3-competition: results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476. The M3- Competition.

Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2018). Statistical and machine learning forecasting methods: Concerns and ways forward. *PLoS ONE*, 13.

Malhotra, P., Vishnu, T., Vig, L., Agarwal, P., and Shroff, G. M. (2017). Timenet: Pre-trained deep recurrent neural network for time series classification. *ArXiv*, abs/1706.08838.

Nichol, A., Achiam, J., and Schulman, J. (2018). On first-order meta-learning algorithms. *ArXiv*, abs/1803.02999.

Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. (2020). N-beats: Neural basis expansion analysis for interpretable time series forecasting. *ArXiv*, abs/1905.10437.

Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. (2021). Meta-learning framework with applications to zero-shot time-series forecasting. *ArXiv*, abs/2002.02887.

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *ArXiv*, abs/1706.03762.

Wang, L.-X. (2015a). Dynamical models of stock prices based on technical trading rules—part ii: Analysis of the model. *IEEE Transactions on Fuzzy Systems*, 23(4):1127–1141.

Wang, L.-X. (2015b). Dynamical models of stock prices based on technical trading rules part i: The models. *IEEE Transactions on Fuzzy Systems*, 23(4):787–801.

Wang, L.-X. (2015c). Dynamical models of stock prices based on technical trading rules—part iii: Application to hong kong stocks. *IEEE Transactions on Fuzzy Systems*, 23(5):1680–1697.

A Appendix

A.1 Model Configurations and Hyperparameters

We use or MLP, LSTM, NBEATS and TFT and data loading and pre-processing utilities. For Transformer, we use the PyTorch's nn.Transformer. Any parameters not included here have been used with their default values as defined in the respective libraries.

We use 5 fast adaptation steps for MAML throughout all the experiments.

A.1.1 Transformer

learning rate	1e-4
batch size	64
weight decay	1e-2
d_{model}	256
attn heads	8
encoders	4
decoders	4
feedforward dim	512
MAML lr	1e-1
learner lr (SN, BN, FR)	1e-5, 1e-3, 1e-4

A.1.2 NBEATS

learning rate	1e-4
batch size	64
weight decay	0
# stacks	1
stack type	generic
backcast loss ratio	0.75
layer width	512
MAML lr	1e-1
learner lr (SN, BN, FR)	1e-3, 1e-3, 1e-3

A.1.3 Temporal Fusion Transformer

learning rate	1e-5
batch size	64
weight decay	0
hidden size	128
# lstm layers	2
# attn heads	4
hidden continuous size	16
# output quantiles	3 (0.1, 0.5, 0.9)
MAML lr	1e-1
learner lr (SN, BN, FR)	1e-4, 1e-3, 1e-3

A.1.4 LSTM

learning rate	1e-4
batch size	64
weight decay	1e-2
# layers	2
hidden state size	10
MAML lr	1e-1
learner lr (SN, BN, FR)	1e-3, 1e-3, 1e-3

A.1.5 MLP

We use hidden layers of uniform size and ReLU activation

learning rate	1e-2
batch size	64
weight decay	0
# hidden layers	3
hidden size	64
MAML lr	1e-1
learner lr (SN, BN, FR)	1e-2, 1e-4, 1e-3