Evolving RNNs for Stock Forecasting: A Low Parameter Efficient Alternative to Transformers

Zimeng Lyu[1], Devroop Kar[1], Matthew Simoni[1], Rohaan Nadeem[1], Avinash Bhojanapalli[1], Hao Zhang[2], and Travis Desell[1]

[1] Golisano College of Computing and Information Sciences,
Rochester Institute of Technology, Rochester NY 14623, USA
[2] Saunders College of Business,
Rochester Institute of Technology, Rochester NY 14623, USA

Abstract. Stock return forecasting is a critical application of time series forecasting in finance, facilitating informed trading and management decisions that can lead to substantial returns. However, for large investment portfolios, designing and fine-tuning models for individual stock predictions is time-consuming and computationally intensive. In this work, we propose using the neuroevolution-based neural architecture search algorithm, Evolutionary eXploration of Augmenting Memory Models (EXAMM), to evolve recurrent neural networks (RNNs) for stock return prediction. We compare the prediction performance of these evolved RNNs with that of state-of-the-art attention-based Transformer and deep learning models. Our results indicate that EXAMM-evolved RNNs outperform or achieve comparable performance with these models across 50 multivariate stock datasets and a combined high-dimensional dataset with 300 input features and 50 outputs. Additionally, they require orders of magnitude fewer parameters and can be evolved and operate efficiently using a minimal 8-core CPU configuration as opposed to expensive GPUs.

Keywords: Recurrent Neural Networks · Time Series Forecasting · Stock Forecasting · Neural Architecture Search · Neuroevolution

1 Introduction

Time series forecasting is an important area of study across various domains, with particular relevance in financial markets. In finance, stock return forecasting is one of the key applications of time series forecasting, where accurate predictions are crucial for making informed decisions in portfolio management and trading. Investment portfolios range in size, from smaller collections of 10-15 stocks to mutual funds holding over 100 stocks. Each stock has its own unique patterns and behaviors, which means effective forecasting models need to capture both the individual characteristics of each stock and any cross-stock dependencies.

Using models for individual stocks can aid in the learning task, however they doing so can limit predictive accuracy as they cannot incorporate inter-stock

dependencies. However, fine-tuning models for a medium-to-large portfolio can require extensive effort. Therefore, designing models that perform well across multiple stocks presents both computational and architectural challenges. Despite recent advancements in machine learning for time series forecasting, most models used for stock return predictions still rely on traditional statistical techniques or simpler, fixed-size models like multi-layer perceptrons (MLPs), long short term memory networks (LSTMs), basic recurrent neural networks (RNNs), or convolutional neural networks (CNNs).

Originally developed for natural language processing, transformers provide a new approach to time series forecasting through their self-attention mechanism, which enables parallel processing and can capture long-range dependencies in temporal data. However, this comes with a tradeoff: transformers have high computational complexity and large parameter sizes, making them resource-intensive, particularly when used for multiple stock forecasting tasks. This limitation is even more apparent when historical data is limited or when computational efficiency is necessary.

On the other hand, RNNs with memory cells like LSTMs [13] and gated recurrent units (GRUs) [4] have been more widely used for time series forecasting due to their sequential processing and efficient parameter usage. These qualities make RNNs more feasible for applications with limited computational resources or moderate datasets, offering a practical balance of simplicity and effectiveness in real-world forecasting tasks.

Neuroevolution-based neural architecture search (NAS) provides an effective strategy for designing and training RNNs specifically for stock return prediction. RNNs can be evolved for each stock individually, allowing them to capture unique patterns without manual tuning. Starting from minimal architectures, neuroevolution can optimize RNNs to achieve strong performance with compact network sizes. By evolving RNNs with various memory cells and deeper recurrent connections, NAS can help address RNNs' limitations in capturing long-term dependencies, without significant manual architectural design effort. Additionally, RNNs can be significantly smaller than transformers, making them computationally efficient for stock return predictions across an entire portfolio.

This study aims to compare RNN and transformer performance for stock return forecasting across a portfolio of 50 stocks. In particular, we investigate two research questions:

- **Research question 1:** Can neuroevolution evolve and train models competitive with modern transformer and deep learning time series forecasting models and if so, are they more efficient in terms of reduced parameter counts and computational needs?
- **Research question 2:** Is it more effective to train and evolve models for stocks individually to provide a simpler learning problem or to combine all stocks together and train a single more complex model which can potentially utilize inter-stock correlations for better forecasts?

To investigate these questions, we utilize the Evolutionary eXploration of Augmenting Memory Models (EXAMM) neuroevolution algorithm to evolve RNNs

for stock return prediction. EXAMM has been well described and examined by prior work [22, 19, 20], however it has not been compared to modern multivariate time series (MTS) forecasting methods, such as transformer models. Results show that utilizing a combined dataset allows models to improve performance by incorporating inter-stock correlation information, and further that models evolved by EXAMM outperform three of the other models (Crossformer, TSMixer and PatchTST) and are competitive with the last (DeformTime) on the combined dataset while requiring orders of magnitude less parameters. Additionally, EXAMM was able to do this utilizing only low cost CPU resources as opposed to significantly more expensive GPU resources.

2 Related Work

Time series stock forecasting employs a variety of methods, ranging from traditional statistical models, such as autoregressive integrated moving average (ARIMA) [29] and autoregressive conditional heteroskedasticity (ARCH) [30], to modern deep learning techniques. Classical statistical models often fail to capture the complex, non-linear dynamics found in stock data. These models generally require the data to be stationary and struggle with data volatility [16]. Machine learning methods address these limitations by utilizing their ability to capture intricate patterns and dependencies [17].

RNNs were among the earliest neural architectures applied to stock price prediction in the financial domain, as demonstrated by Kamijo *et al.* and Kiani *et al.* [14][15]. However RNN models often encounter challenges such as vanishing gradients and fail to retain information over long sequences. Advanced memory cells LSTM and gated recurrent units (GRU) have been integrated into RNN models to capture long-term dependencies effectively. LSTM-RNN models have been extensively employed for predicting stock prices and market trends, as shown in studies by Smith *et al.*, Gao *et al.*, and Ghosh *et al.* [32, 9, 10]. Additional research has further validated the versatility of LSTM-RNNs in forecasting stock movements under varying market conditions [43, 39, 28]. Similarly, GRU-RNNs have been successfully applied to stock and market predictions [3, 12]. Beyond single architectures, Zhang *et al.* demonstrated the efficacy of integrating deep belief networks (DBN) with LSTMs for stock price movement prediction [41]. Hybrid RNN-CNN architectures, which combine the temporal modeling strengths of RNNs with the feature extraction capabilities of CNNs, have further enhanced predictive accuracy, as explored by Zhang *et al.* and Yang *et al.* [40, 37].

Transformers, originally developed for neural machine translation [34], have been adapted for multivariate time series forecasting and have shown great potential in capturing temporal and cross-dimensional dependencies. Crossformer utilizes Dimension-Segment-Wise (DSW) embeddings and a Two-Stage Attention (TSA) layer to model cross-time and cross-dimension patterns effectively [42]. Similarly, DeformTime introduces deformable attention blocks (DABs) to enhance temporal pattern recognition across variables [31]. Lightweight alternatives

like TSMixer adopt a Multi-Layer Perceptron (MLP)-based approach inspired by MLP-Mixer models, incorporating reconciliation heads and gated attention to reduce computational overhead [35]. PatchTST further optimizes resource usage and accuracy by independently processing each channel with a patching mechanism, excelling in long-term predictions through self-supervised pre-training [24].

Bio-inspired methodologies present novel approaches to time series forecasting through evolution-based optimization and neural architecture search. Basterrech *et al.* designed EvoESN, merging evolutionary algorithms with Echo State Networks to evolve reservoir structures in Fourier space [1]. Lyu *et al.* introduced ONE-NAS, which is capable of utilizing neuroevolution to evolve RNN architectures in online scenarios [21]. For financial crisis prediction, Uthayakumar *et al.* developed Ant Colony Optimization financial crisis prediction (ACO-FCP) for feature selection and classification [33]. Yang *et al.* combined Artificial Bee Colony optimization with Extreme Learning Machine for short-term traffic flow analysis [38]. Breaking from more conventional approaches, Elsaid *et al.* developed a continuous ant-based topology search to optimize neural architectures without backpropagation, delivering competitive results with reduced computational demands [7].

3 EXAMM

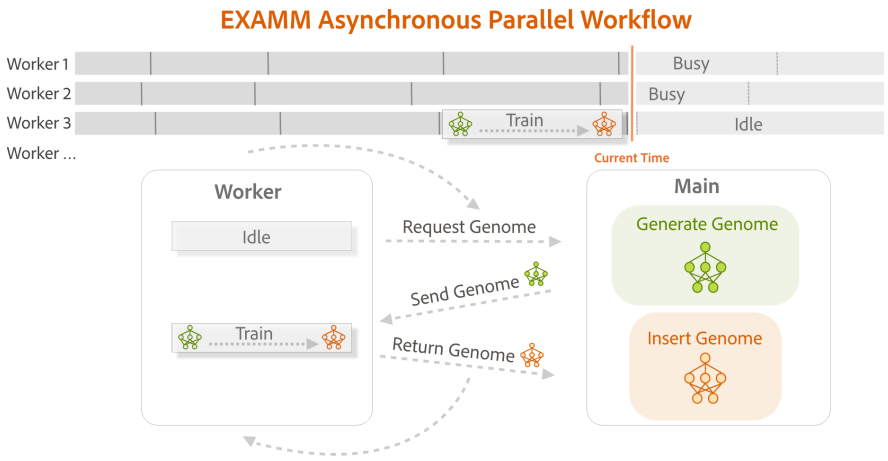

Fig. 1. EXAMM uses a asynchronous strategy allowing for it to be easily distributed across multiple processes and automatically load balanced.

In comparison with other work, this work utilizes the Evolutionary eXploration of Augmenting Memory Models (EXAMM) neuroevolution algorithm [25] to evolve RNNs to make stock return forecasts. EXAMM evolves progressively

larger RNNs through a series of mutation and crossover (reproduction) operations from minimal seed networks that only contain input to output connections. Newly generated neurons in these genomes are selected from a library of neurons with traditional activation functions, Δ-RNN units [26], gated recurrent units (GRUs) [4], long short-term memory cells (LSTMs) [13], minimal gated units (MGUs) [44], and update-gate RNN cells (UGRNNs) [5]. Newly generated edges can be deeply recurrent, spanning multiple time steps, which has been found to significantly improve predictive ability [6]. Weights of newly generated genomes are inherited from their parents to significantly reduce the amount of training needed and to reveal more performant areas of the RNN training search space [19].

EXAMM's population of RNNs is distributed across distinct islands where genomes in each island are generated through 11 types of mutation, intra-island crossover island crossover to preserve gene diversity across islands, and more periodically inter-island crossover to for information exchange across islands [25]. Islands are repopulated periodically to prevent the islands from becoming stagnant and to further improve performance [20].

Figure 1 illustrates EXAMM's asynchronous parallel workflow for training newly generated genomes. In this parallel system, the main process is responsible for generating and inserting trained genomes while managing the entire population. Worker processes are dedicated to training these newly generated genomes. When a worker becomes idle, it sends a request to the main process, which then generates a new genome (through the possible mutation or crossover methods) and dispatches it to the worker for training. After training, the worker sends the trained genome back to the main process, along with a new work request. This workflow optimizes the utilization of worker processes for training and scales easily based on computational capacity. In this strategy no worker waits on any other worker to compute a new genome, which is especially important as the generated RNN genomes take varying amounts of time to train based on their size. Notably, this algorithm operates solely on CPUs, making it more affordable to scale. This allows medium to small datasets to run efficiently even on a personal laptop, eliminating the need for GPU desktops, clouds or clusters.

4 CRSP Dataset

The Center for Research in Security Prices (CRSP) dataset is a comprehensive collection of data on the U.S. stock market. It contains all stocks on the NYSE, AMEX, and NASDAQ, and covers wide range of market indexes, open-end mutual funds, exchange-traded funds (ETFs), U.S. Treasury securities, and corporate bonds. CRSP data is known for its high level of accuracy and is highly reliable for academic research and financial modeling [2, 18]. CRSP data has a long historical range from December 1925 to present, with daily, monthly, quarterly, and annual data. The dataset contains valuable variables for financial research such as stock prices, returns, trading volumes, and dividend yields.

4.1 Stock Selection

For this work, 50 mid-cap companies[3] were selected from the S&P 500 index to forecast stock returns with the goal of making accurate predictions that could support investment decisions aiming to outperform the market. Mid- and small-cap stocks present a unique opportunity for return forecasting due to their lower market efficiency compared to large-cap stocks. Unlike large-cap stocks, which are extensively analyzed by institutional investors and market analysts, mid- and small-caps receive less scrutiny, meaning their prices do not always reflect new information immediately. This lower efficiency allows potential patterns and inefficiencies to exist within mid- and small-cap stock prices, providing opportunities for excess returns and better forecasting. Given that the S&P 500 index primarily includes large- and mid-cap companies in the U.S. market, we focused on 50 mid-cap companies to conduct our return forecasting experiments.

4.2 Predictors

From this dataset, we chose 6 economic predictors that are associated with stock returns, shown in Table 1. Those predictors were directly obtained from the CRSP dataset or calculated by using variables form the CRSP dataset.

Table 1. Economic Predictors for Stock Return Prediction

Predictors	Description
Return	Percentage change in stock price
Volume Change	Percentage change in trading volume
Bid-Ask Spread	$(AskPrice - BidPrice)/Price$
Illiquidity	$Return/(Volume \times Price)$
Turn Over	$Volume/SharesOutstanding$
S&P 500 Return	S&P 500 index return

Return is the percentage change of a stock's price from time $[t-1]$ to time $[t]$, which in the case of this data is day-to-day. *Return* was also used as the forecasting output parameter. *Volume Change* is the percentage change of the stock's trading volume from time $[t-1]$ to time $[t]$. *Bid-Ask Spread* measures the difference between the highest price buyers are willing to pay (Bid) and the lowest price the sellers are willing to sell (Ask), with the Ask price usually being higher than the *Bid* price. *Illiquidity* measures if the stock can easily be sold for cash. *Turn Over* measures the rate of a stock being sold during a given period, and it is a measure of stock liquidity. *S&P 500 Return* is the return of the entire S&P 500 index, and it often serve as an overall benchmark for market performance. These predictors contain information for stock performance as well as market-wide information, and have been shown to be associated with stock returns [8, 23, 11].

[3] A mid-cap company is a company with a market capitalization between $2 billion and $10 billion.

4.3 Experimental Stock Datasets

We first created 50 individual stock datasets[4], with each containing the six economic predictors, with *Return* as the target output. We utilized all available historical data for each stock from the CRSP database, resulting in varied starting dates. Specifically, six of the stocks have data starting after 2000, while the remaining stocks begin before 2000, with all datasets ending in 2023. After calculating predictors and cleaning rows with missing information, the average dataset length across all the stocks was 7,333 rows.

Recognizing that market information impacts individual stock performance beyond index values like the S&P 500 returns, we explored the potential for models to learn inter-stock information. To facilitate this and to evaluate the performance of EXAMM and the transformer models on large datasets, we horizontally combined all 50 stock datasets into a single, unified dataset. This combined dataset includes all predictors from each stock as input parameters and each stock's return as the output, resulting in 300 columns (6 per stock) and 4,063 rows, with data starting from the earliest time that data was available for all 50 stocks.

5 Results

5.1 Experimental Setup

This experiment compares the models evolved by EXAMM with three fixed transformer models, Crossformer [42], DeformTime [31] and PatchTST [24]; and a modern deep linear model, TSMixer [35], across the 50 stock datasets and the large scale dataset combining all 50 stocks. These fixed models were chosen as they considered to be the most recent widely cited state of the art models for time series forecasting. These models have shown the best performance on a series of benchmark problems, outperforming older state of the art models such as Autoformer [36] and Informer [45]. The two sets of experiments, individual stocks and a combined dataset of all stocks, were chosen to determine if better forecasts could be achieved by training models on simpler tasks - one stock at a time, or on the large complex task of all stocks which could potentially enable learning correlation between stocks, at the cost of a much more challenging training and neuroevolution problem.

For each stock dataset, 20 repeated experiments were performed with EXAMM and each fixed model, resulting in a total of 1,020 experiment runs per model/methodology (1000 runs for individual stocks, with an additional 20 for the combined data). All models, including EXAMM, were evaluated using their default hyperparameters without additional tuning.

Each EXAMM run used 10 islands, each with a maximum capacity of 10 genomes. Island repopulation was employed to prevent stagnation and improve performance, with repopulation occurring every 500 generated genomes [20]. A

[4] https://github.com/zimenglyu/CRSP_Processor

Table 2. Best MAE

Stock	EXAMM	Crossfomer	DeformTime	TSMixer	PatchTST
AKAM	0.01195	0.01325	**0.01159**	0.03683	0.07359
ATO	0.01082	**0.01069**	0.01077	0.01739	0.18505
BXP	0.01847	0.01560	0.01849	0.01801	**0.01187**
CAG	0.01151	0.01254	**0.01151**	0.03861	0.06947
CHRW	0.01295	**0.01286**	0.01293	0.01427	0.11019
CINF	0.01430	0.01407	0.01432	**0.01387**	0.11187
COO	0.01293	0.01310	**0.01292**	0.01854	0.09730
CPB	0.01076	0.01125	**0.01075**	0.01278	0.06372
CPT	0.01267	**0.01203**	0.01267	0.01292	0.19492
DECK	**0.01936**	0.02022	0.01940	0.02707	0.62254
DPZ	0.01330	0.01325	**0.01318**	0.03553	0.24503
DVA	0.01537	0.01542	**0.01455**	0.01670	0.23211
ED	0.01024	**0.01018**	0.01020	0.01382	0.07648
EIX	**0.01305**	0.01341	0.01313	0.02632	0.30901
EMN	**0.01624**	0.01638	0.01625	0.01753	0.07363
ESS	0.01369	**0.01297**	0.01362	0.02100	0.04408
ETR	0.01174	**0.01139**	0.01175	0.02834	0.16368
EXPD	0.01202	0.01194	**0.01184**	0.01545	0.07939
EXR	**0.01313**	0.01338	0.01314	0.02012	0.19506
FDS	0.01271	0.01296	**0.01265**	0.01746	0.15597
FFIV	0.01370	0.01537	**0.01368**	0.02567	0.21021
FRT	0.01486	**0.01446**	0.01486	0.01571	0.12558
HBAN	0.01737	**0.01646**	0.01736	0.01823	0.26058
HOLX	0.01275	0.01320	**0.01275**	0.03139	0.36762
HSIC	0.01332	0.01371	**0.01332**	0.02611	0.17979
IEX	**0.01180**	0.01200	0.01183	0.01386	0.10627
IRM	0.01436	0.01503	**0.01345**	0.01765	0.17098
IVZ	0.01947	0.01968	**0.01791**	0.04191	0.21389
JBHT	0.01417	**0.01400**	0.01414	0.01685	0.20124
JKHY	0.01168	**0.01133**	0.01168	0.02810	0.12262
KIM	0.01739	0.01716	0.01743	**0.01594**	0.21076
KMB	0.00934	0.00993	**0.00932**	0.00978	0.07880
KMX	0.01922	**0.01900**	0.02046	0.02319	0.08841
LH	**0.01268**	0.01352	0.01305	0.01559	0.31573
LNT	**0.01095**	0.01130	0.01097	0.01125	0.14147
MHK	0.02090	0.02173	0.02191	0.02400	**0.01042**
NDSN	0.01291	0.01316	**0.01289**	0.01457	0.20885
NTRS	0.01546	**0.01469**	0.01519	0.02163	0.14724
NVR	0.01575	0.01620	**0.01567**	0.01748	0.13200
PKG	**0.01186**	0.01417	0.01187	0.01816	0.07292
PODD	0.02118	0.02115	**0.02013**	0.03052	0.07870
REG	0.01660	0.01483	0.01516	**0.01216**	0.15057
RHI	0.01561	0.01560	**0.01535**	0.01971	0.13566
STLD	**0.02091**	0.02103	0.02092	0.02442	0.07325
TECH	0.01618	**0.01594**	0.01614	0.02760	0.09670
TFX	0.01504	0.01497	0.01494	**0.01178**	0.06951
TRMB	0.01674	**0.01648**	0.01659	0.02337	0.15758
TRV	0.01178	0.01120	**0.01019**	0.01526	0.07934
UHS	**0.01562**	0.01589	0.01584	0.02806	0.14717
WRB	0.01215	0.01185	**0.01107**	0.02214	0.22296
Sum	0.71896	0.72194	0.71173	1.04465	7.69175
Count	10	14	20	4	2

Table 3. Best MSE

Stock	EXAMM	Crossfomer	DeformTime	TSMixer	PatchTST
AKAM	3.00e-4	3.73e-4	**2.88e-4**	2.72e-3	5.42e-3
ATO	2.71e-4	2.71e-4	**2.70e-4**	1.04e-3	3.42e-2
BXP	6.89e-4	5.47e-4	6.90e-4	6.92e-4	**1.41e-4**
CAG	**2.92e-4**	3.75e-4	2.93e-4	2.08e-3	4.83e-3
CHRW	3.39e-4	3.54e-4	**3.38e-4**	4.09e-4	1.21e-2
CINF	4.86e-4	**4.77e-4**	4.87e-4	5.83e-4	1.25e-2
COO	**3.47e-4**	3.68e-4	3.48e-4	9.13e-4	9.47e-3
CPB	**2.45e-4**	2.71e-4	2.46e-4	2.95e-4	4.06e-3
CPT	3.57e-4	**3.39e-4**	3.58e-4	3.71e-4	3.80e-2
DECK	**7.71e-4**	8.17e-4	7.71e-4	2.16e-3	3.88e-1
DPZ	3.73e-4	**3.50e-4**	3.64e-4	2.30e-3	6.00e-2
DVA	5.67e-4	5.71e-4	5.09e-4	**4.82e-4**	5.39e-2
ED	2.51e-4	2.54e-4	**2.50e-4**	3.62e-4	5.85e-3
EIX	**3.55e-4**	3.86e-4	3.59e-4	1.19e-3	9.55e-2
EMN	**5.22e-4**	5.32e-4	5.22e-4	5.98e-4	5.42e-3
ESS	4.05e-4	**3.83e-4**	4.04e-4	4.72e-4	1.94e-3
ETR	3.23e-4	**3.12e-4**	3.24e-4	1.30e-3	2.68e-2
EXPD	2.91e-4	2.88e-4	**2.79e-4**	5.40e-4	6.30e-3
EXR	**3.48e-4**	3.50e-4	3.50e-4	7.38e-4	3.80e-2
FDS	**3.66e-4**	3.88e-4	3.66e-4	5.70e-4	2.43e-2
FFIV	3.66e-4	4.66e-4	**3.65e-4**	1.34e-3	4.42e-2
FRT	5.77e-4	5.70e-4	5.78e-4	**5.70e-4**	1.58e-2
HBAN	6.55e-4	**5.99e-4**	6.52e-4	9.77e-4	6.79e-2
HOLX	**3.56e-4**	3.74e-4	3.56e-4	1.81e-3	1.35e-1
HSIC	**3.66e-4**	3.78e-4	3.66e-4	1.19e-3	3.23e-2
IEX	2.81e-4	2.88e-4	**2.78e-4**	3.44e-4	1.13e-2
IRM	4.50e-4	5.01e-4	**3.60e-4**	5.87e-4	2.92e-2
IVZ	6.59e-4	6.74e-4	**5.63e-4**	2.99e-3	4.57e-2
JBHT	3.90e-4	**3.83e-4**	3.88e-4	6.05e-4	4.05e-2
JKHY	2.93e-4	**2.73e-4**	2.95e-4	1.93e-3	1.50e-2
KIM	7.45e-4	7.46e-4	7.46e-4	**5.78e-4**	4.44e-2
KMB	1.91e-4	2.16e-4	**1.90e-4**	2.01e-4	6.21e-3
KMX	**7.33e-4**	8.13e-4	9.03e-4	1.20e-3	7.82e-3
LH	**3.89e-4**	4.33e-4	4.13e-4	6.41e-4	9.97e-2
LNT	2.56e-4	2.81e-4	**2.17e-4**	2.73e-4	2.00e-2
MHK	9.49e-4	1.06e-3	1.03e-3	1.08e-3	**1.09e-4**
NDSN	**3.89e-4**	3.94e-4	3.89e-4	4.87e-4	4.36e-2
NTRS	5.01e-4	**4.58e-4**	4.92e-4	9.19e-4	2.17e-2
NVR	5.84e-4	6.02e-4	**5.76e-4**	9.48e-4	1.74e-2
PKG	**2.65e-4**	4.16e-4	2.66e-4	1.38e-3	5.32e-3
PODD	8.83e-4	8.85e-4	**8.44e-4**	2.34e-3	6.19e-3
REG	6.77e-4	6.25e-4	6.23e-4	**4.03e-4**	2.27e-2
RHI	4.95e-4	5.17e-4	**4.87e-4**	7.54e-4	1.84e-2
STLD	**8.35e-4**	8.60e-4	8.59e-4	1.14e-3	5.37e-3
TECH	4.79e-4	**4.75e-4**	4.80e-4	1.92e-3	9.35e-3
TFX	4.56e-4	4.48e-4	4.52e-4	**3.04e-4**	4.83e-3
TRMB	5.82e-4	5.82e-4	**5.69e-4**	1.15e-3	2.48e-2
TRV	3.28e-4	2.16e-4	**1.83e-4**	4.32e-4	6.29e-3
UHS	**6.30e-4**	6.61e-4	6.51e-4	1.97e-3	2.17e-2
WRB	3.35e-4	2.45e-4	**2.17e-4**	7.35e-4	4.97e-2
Sum	2.30e-2	2.34e-2	2.26e-2	5.10e-2	1.70
Count	16	10	17	5	2

Table 4. Parameter Count

Stock	EXAMM	Crossfomer	DeformTime	TSMixer	PatchTST
AKAM	**119**	11302693	395285	548	17476
ATO	**134**	11302693	395285	548	17476
BXP	**110**	11302693	395285	548	17476
CAG	**117**	11302693	395285	548	17476
CHRW	**146**	11302693	395285	548	17476
CINF	**110**	11302693	395285	548	17476
COO	**98**	11302693	395285	548	17476
CPB	**103**	11302693	395285	548	17476
CPT	**123**	11302693	395285	548	17476
DECK	**121**	11302693	395285	548	17476
DPZ	**148**	11302693	395285	548	17476
DVA	**114**	11302693	395285	548	17476
ED	**129**	11302693	395285	548	17476
EIX	**188**	11302693	395285	548	17476
EMN	**127**	11302693	395285	548	17476
ESS	**110**	11302693	395285	548	17476
ETR	**139**	11302693	395285	548	17476
EXPD	**103**	11302693	395285	548	17476
EXR	**128**	11302693	395285	548	17476
FDS	**146**	11302693	395285	548	17476
FFIV	**120**	11302693	395285	548	17476
FRT	**111**	11302693	395285	548	17476
HBAN	**122**	11302693	395285	548	17476
HOLX	**127**	11302693	395285	548	17476
HSIC	**122**	11302693	395285	548	17476
IEX	**141**	11302693	395285	548	17476
IRM	**118**	11302693	395285	548	17476
IVZ	**138**	11302693	395285	548	17476
JBHT	**103**	11302693	395285	548	17476
JKHY	**114**	11302693	395285	548	17476
KIM	**120**	11302693	395285	548	17476
KMB	**117**	11302693	395285	548	17476
KMX	**143**	11302693	395285	548	17476
LH	**132**	11302693	395285	548	17476
LNT	**115**	11302693	395285	548	17476
MHK	**121**	11302693	395285	548	17476
NDSN	**116**	11302693	395285	548	17476
NTRS	**106**	11302693	395285	548	17476
NVR	**155**	11302693	395285	548	17476
PKG	**137**	11302693	395285	548	17476
PODD	**138**	11302693	395285	548	17476
REG	**144**	11302693	395285	548	17476
RHI	**123**	11302693	395285	548	17476
STLD	**127**	11302693	395285	548	17476
TECH	**122**	11302693	395285	548	17476
TFX	**109**	11302693	395285	548	17476
TRMB	**138**	11302693	395285	548	17476
TRV	**109**	11302693	395285	548	17476
UHS	**105**	11302693	395285	548	17476
WRB	**120**	11302693	395285	548	17476
Sum	**6227**	565134650	19764250	27400	873800

total of $10k$ genomes were generated in each experiment. Recurrent connections were allowed to span any time-skip randomly selected from $\mathcal{U}(1, 10)$. Backpropagation through time (BPTT) was performed with a learning rate of $\eta = 0.001$ and used Adam momentum with $\mu = 0.9$. Each generated RNN was trained for 10 BP epochs. Gradient boosting and gradient scaling [27] were applied when the gradient norm fell below 0.05 or exceeded 1.0.

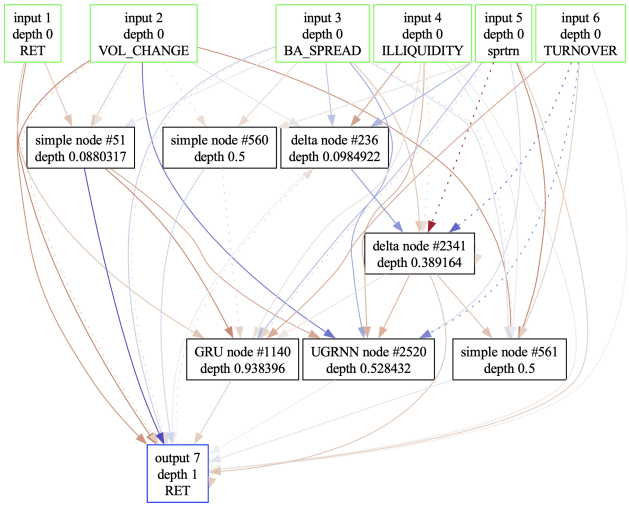

Fig. 2. Example of an EXAMM Evolved RNN for forecasting the CAG stock return.

Hyperparameters used for the fixed models are provided in Appendix A in the supplementary material. The final best trained RNNs by EXAMM are all provided in the supplementary material along with instructions for re-running them over the data for reproducibility. As an example, Figure 2 shows the EXAMM evolved RNN model for CAG stock, where darker blue lines show connections with more positive weight values, darker red lines show connections with more negative weight values, and dotted lines represent recurrent connections.

All datasets used a 70/15/15 percent split for training, validation, and testing. EXAMM was run with MPI using 8 CPU cores per experiment, on a research computing environment with Intel® Xeon® Gold 6150 2.70GHz cores. The fixed models required GPU configurations and those experiments done using NVidia A100 GPUs. *This is notable as EXAMM is able to evolve RNNs for time series forecasting without requiring expensive GPU resources.*

5.2 Individual Stock Forecasting

Tables 2 and 3 present the best MAE and MSE of each model on the test data across all 50 stocks, with bold indicating the best performance for each stock

across all models. In portfolio management and stock trading, it is essential to evaluate the predictive accuracy not only for individual stocks but also for the portfolio as a whole. To facilitate this comparison, we summarize and present the overall MSE/MAE values for each model across all 50 stocks, with the cumulative values displayed at the bottom of each table. Additionally, each table includes a count of best performing cases over 50 stocks of each model; e.g., EXAMM achieved the top performance on 10 stocks by MAE, and 16 stocks by MSE.

In both tables, EXAMM's performance is comparable to the top two performing transformer models—Crossformer and DeformTime. For most stocks, the predictive accuracy of EXAMM is very close to that of the leading transformer models. In terms of the overall sum of MSE/MAE, EXAMM closely approaches DeformTime, the best-performing model. These results suggest that EXAMM's evolved RNNs hold strong potential for time series forecasting, achieving competitive performance with transformer-based approaches.

5.3 Combined Stock Forecasting

Table 5. Combined Dataset Results

	EXAMM	Fine Tuned	Crossfomer	DeformTime	TSMixer	PatchTST
Best MAE	0.73902	0.70799	0.72581	0.66872	0.72293	8.36444
Best MSE	0.02058	0.01923	0.01983	0.01756	0.02340	3.19147
# Params	16703	16703	13455842	405365	207645	17476

Table 6. Combined Individual Model Results

	EXAMM	Crossfomer	DeformTime	TSMixer	PatchTST
50 stock MAE Sum	0.71896	0.72194	0.71173	1.04465	7.69175
50 stock MSE Sum	0.02299	0.02344	0.02261	0.05100	1.69894
50 stock # Params	6227	565134650	19764250	27400	873800

Table 5 shows EXAMM and the fixed models' performance on the combined stock dataset, where a single model predicts all stock returns and Table 6 shows EXAMM and the fixed models' combined performance where 50 models each predicted a single stock return. To be comparable, the results in both Tables present the sum of errors over 50 stocks as opposed to the overall MSE and MAE over all outputs. The *Fine Tuned* column in Table 5 shows additional results where the best found RNN evolved by EXAMM was fine tuned by being trained for an additional 100 epochs, after reducing the learning rate to 0.0005. It is important to note that while EXAMM's Lamarckian weight inheritance

strategy reduces the number of epochs required for training the neural networks during the neuroevolution process, each RNN generated was still only trained for 10 epochs. This does leave some additional room for fine tuning the final best genome for additional peformance.

Results show that before fine tuning, the evolved RNNs have performance comparable to TSMixer, Crossformer and DeformTime, while having significantly less number of trainable parameters. After fine tuning, the best EXAMM RNN outperformed all models other than DeformTime. In particular, we find it particularly impressive that EXAMM was to evolve a competitive model for a very complicated time series forecasting (300 inputs and 50 outputs) from a minimal seed genome with only edge and node level mutation operations alongside crossover. Additionally, it was able to find such an architecture with a fairly low number of generated RNN genomes (10k) given the complexity of the problem.

Table 6 addresses the research question of if it was better to train individual models for each stock or an overall combined model. When comparing the sum of 50 stock models' MAE/MSE, we found that having additional information to determine correlations between the various stocks was able to provide improvement over the added complexity of having one more complex model. Similarly to the combined model, we found that for predicting stocks with individual models, EXAMM outperformed all the other models except for DeformTime.

5.4 Computational Efficiency and Model Complexity

Table 4 presents the average trainable parameter count for each model. As EXAMM evolves RNNs through evolutionary mutations and crossovers, its trainable parameter count varies across repeated experiments but remains within a fairly similar range. This table highlights the compactness of the EXAMM-evolved RNNs compared to the significantly larger fixed models. For models with similar predictive performance to EXAMM on the combined dataset—Crossformer, DeformTime and TSMixer—EXAMM's parameter count is substantially smaller, with 800 times fewer parameters than Crossformer and 24.2 times fewer than DeformTime and 12.4 times fewer than TSMixer. These findings underscore EXAMM's potential in evolving small yet highly effcient and effective RNNs for time series forecasting across diverse datasets. For the EXAMM algorithm on the given hardware, each experiment on the single single stock datasets only took between 5 and 10 minutes using 8 compute cores of CPU resources, and 3-4 days using 8 compute cores of CPU resources for the combined model. It should be noted that EXAMM is easily scalable with MPI so the compute time, especially for the combined model, can be significantly reduced by adding more compute cores, however for this work less cores were utilized due to performing 20 experiments simultaneously on the compute cluster.

6 Conclusion

This work investigates the time series forecasting performance of RNNs and transformers in predicting the returns of 50 stocks for portfolio trading and

management applications. While transformer models represent a more recent advancement in time series forecasting, they generally require more computational resources than RNNs, making them potentially less suitable for predicting stock returns in large trading portfolios. The RNNs in this study were evolved through EXAMM, a distributed asynchronous neuroevolution NAS. EXAMM evolves progressively larger RNNs from minimal seed networks, optimizing outcomes with smaller networks. Memory cells and deep recurrent structures enable RNNs to capture long-term dependencies, potentially achieving similar performance to state-of-the-art time series forecasting models.

We compared time series forecasting models evolved by EXAMM with state of the art transformer and deep learning models (Crossformer [42], DeformTime [31], and PatchTST [24], and TSMixer [35]) in order to address research questions related to the effectiveness of training individual models per stock as opposed to larger models forecasting multiple stocks simultaneously, as well has to determine the potential efficiency and accuracy improvements from utilizing neuroevolution. We found that across all models it was more effective to train a single model across all stocks, as the models were able to gain predictive information due to correlations between stock parameters. Additionally, we found that EXAMM was able to evolve models that outperformed Crossformer, TSMixer and PatchTST while requiring 800, 12.4 and 1.05 times fewer parameters (EXAMM's models were also orders of magnitude more peformant than PatchTST which performed very poorly on the task). It also performed competitively with DeformTime with 24.2 times fewer parameters. EXAMM was also able to accomplish this only utilizing low cost CPU resources as opposed to expensive GPUs required by the transformer and deep learning models.

Although some research has proposed light weight transformer models for effective time series forecasting, it is noteworthy that RNNs designed through NAS can achieve performance comparable to large attention-based transformer models while remaining significantly more parameter-efficient. This finding shows that well designed RNNs can still perform well on real world time series forecasting tasks despite the recent dominance of transformer models in time series forecasting. Our results show that neuroevolution can effectively design and train RNNs for time series forecasting of stock data, even on complex multi-stock forecasting problems generating models significantly more efficient than transformer and deep learning methods. Further, this research presents avenues for further improving the performance of EXAMM. In particular, especially for the combined stock dataset EXAMM was only run for a relatively small number of RNN genomes (10k). With a longer evolution time and further hyperparameter adjustment, the models could potentially even outperform DeformTime. Additionally, DeformTime's deformable attention blocks (DABs) seem to provide significant benefit over the other time series forecasting methods evaluated. Incorporating variable and temporal DAB elements into EXAMM could lead to even better performing, yet efficient forecasting models.

References

1. Basterrech, S., Rubino, G.: Evolutionary echo state network: A neuroevolutionary framework for time series prediction. Applied Soft Computing **144**, 110463 (2023)
2. Center for Research in Security Prices (CRSP): CRSP US Stock Databases. The University of Chicago Booth School of Business (2023), http://www.crsp.com, Data set
3. Chen, C., Xue, L., Xing, W.: Research on improved gru-based stock price prediction method. Applied Sciences **13**(15), 8813 (2023)
4. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
5. Collins, J., Sohl-Dickstein, J., Sussillo, D.: Capacity and trainability in recurrent neural networks. arXiv preprint arXiv:1611.09913 (2016)
6. Desell, T., ElSaid, A., Ororbia, A.G.: An empirical exploration of deep recurrent connections using neuro-evolution. In: The 23nd International Conference on the Applications of Evolutionary Computation (EvoStar: EvoApps 2020). Seville, Spain (April 2020)
7. ElSaid, A., Ricanek, K., Lyu, Z., Ororbia, A., Desell, T.: Backpropagation-free 4d continuous ant-based neural topology search. Applied Soft Computing **147**, 110737 (2023)
8. Foucault, T., Pagano, M., Röell, A.: Market liquidity: theory, evidence, and policy. Oxford University Press, USA (2013)
9. Gao, S.E., Lin, B.S., Wang, C.M.: Share price trend prediction using crnn with lstm structure. In: 2018 International Symposium on Computer, Consumer and Control (IS3C). pp. 10–13. IEEE (2018)
10. Ghosh, A., Bose, S., Maji, G., Debnath, N., Sen, S.: Stock price prediction using lstm on indian share market. In: Proceedings of 32nd international conference on. vol. 63, pp. 101–110 (2019)
11. Grimes, D.: Stock market logic: a sophisticated approach to profits on wall street (1977)
12. Gupta, U., Bhattacharjee, V., Bishnu, P.S.: Stocknet—gru based stock index prediction. Expert Systems with Applications **207**, 117986 (2022)
13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
14. Kamijo, K.i., Tanigawa, T.: Stock price pattern recognition-a recurrent neural network approach. In: 1990 IJCNN international joint conference on neural networks. pp. 215–221. IEEE (1990)
15. Kiani, K.M., Kastens, T.L.: Testing forecast accuracy of foreign exchange rates: Predictions from feed forward and various recurrent neural network architectures. Computational Economics **32**, 383–406 (2008)
16. Kumbure, M.M., Lohrmann, C., Luukka, P., Porras, J.: Machine learning techniques and data for stock market forecasting: A literature review. Expert Systems with Applications **197**, 116659 (2022)
17. Leippold, M., Wang, Q., Zhou, W.: Machine learning in the chinese stock market. Journal of Financial Economics **145**(2), 64–82 (2022)
18. Lemieux, V., Rahmdel, P.S., Walker, R., Wong, B.W., Flood, M.: Clustering techniques and their effect on portfolio formation and risk analysis. In: Proceedings of the International Workshop on Data Science for Macro-Modeling. pp. 1–6 (2014)

19. Lyu, Z., ElSaid, A., Karns, J., Mkaouer, M., Desell, T.: An experimental study of weight initialization and lamarckian inheritance on neuroevolution. The 24th International Conference on the Applications of Evolutionary Computation (EvoStar: EvoApps) (2021)
20. Lyu, Z., Karnas, J., ElSaid, A., Mkaouer, M., Desell, T.: Improving distributed neuroevolution using island extinction and repopulation. The 24th International Conference on the Applications of Evolutionary Computation (EvoStar: EvoApps) (2021)
21. Lyu, Z., Ororbia, A., Desell, T.: Online evolutionary neural architecture search for multivariate non-stationary time series forecasting. Applied Soft Computing **145**, 110522 (2023)
22. Lyu, Z., Patwardhan, S., Stadem, D., Langfeld, J., Benson, S., Thoelke, S., Desell, T.: Neuroevolution of recurrent neural networks for time series forecasting of coal-fired power plant operating parameters. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 1735–1743 (2021)
23. Malinova, K., Park, A.: Liquidity, volume, and price behavior: The impact of order vs. quote based trading. In: 11th Symposium on Finance, Banking, and Insurance, University of Karlsruhe, December. Citeseer (2008)
24. Nie, Y., Nguyen, N.H., Sinthong, P., Kalagnanam, J.: A time series is worth 64 words: Long-term forecasting with transformers. arXiv preprint arXiv:2211.14730 (2022)
25. Ororbia, A., ElSaid, A., Desell, T.: Investigating recurrent neural network memory structures using neuro-evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 446–455. GECCO '19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3321707.3321795, http://doi.acm.org/10.1145/3321707.3321795
26. Ororbia II, A.G., Mikolov, T., Reitter, D.: Learning simpler language models with the differential state framework. Neural Computation **0**(0), 1–26 (2017). https://doi.org/10.1162/neco_a_01017, https://doi.org/10.1162/neco_a_01017, pMID: 28957029
27. Pascanu, R., Mikolov, T., Bengio, Y.: On the difficulty of training recurrent neural networks. In: International Conference on Machine Learning. pp. 1310–1318 (2013)
28. Pawar, K., Jalem, R.S., Tiwari, V.: Stock market price prediction using lstm rnn. In: Emerging Trends in Expert Applications and Security: Proceedings of ICETEAS 2018. pp. 493–503. Springer (2019)
29. Reddy, C.V.: Predicting the stock market index using stochastic time series arima modelling: The sample of bse and nse. Indian Journal of Finance **13**(8), 7–25 (2019)
30. Selvin, S., Vinayakumar, R., Gopalakrishnan, E.A., Menon, V.K., Soman, K.P.: Stock price prediction using lstm, rnn and cnn-sliding window model. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). pp. 1643–1647 (2017). https://doi.org/10.1109/ICACCI.2017.8126078
31. Shu, Y., Lampos, V.: Deformtime: Capturing variable dependencies with deformable attention for time series forecasting. arXiv preprint arXiv:2406.07438 (2024)
32. Smith, N., Varadharajan, V., Kalla, D., Kumar, G.R., Samaah, F.: Stock closing price and trend prediction with lstm-rnn. Journal of Artificial Intelligence and Big Data pp. 1–13 (2024)
33. Uthayakumar, J., Metawa, N., Shankar, K., Lakshmanaprabu, S.: Financial crisis prediction model using ant colony optimization. International Journal of Information Management **50**, 538–556 (2020)

34. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
35. Vijay, E., Jati, A., Nguyen, N., Sinthong, G., Kalagnanam, J.: Tsmixer: Lightweight mlp-mixer model for multivariate time series forecasting. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2023)
36. Wu, H., Xu, J., Wang, J., Long, M.: Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. Advances in neural information processing systems **34**, 22419–22430 (2021)
37. Yang, J., Zhang, M., Fang, R., Zhang, W., Zhou, J.: Separating the predictable part of returns with cnn-gru-attention from inputs to predict stock returns. Applied Soft Computing **165**, 112116 (2024)
38. Yang, Y., Duan, Z.: An effective co-evolutionary algorithm based on artificial bee colony and differential evolution for time series predicting optimization. Complex & Intelligent Systems **6**(2), 299–308 (2020)
39. Yao, S., Luo, L., Peng, H.: High-frequency stock trend forecast using lstm model. In: 2018 13th International Conference on Computer Science & Education (ICCSE). pp. 1–4. IEEE (2018)
40. Zhang, R., Yuan, Z., Shao, X.: A new combined cnn-rnn model for sector stock price analysis. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). vol. 2, pp. 546–551. IEEE (2018)
41. Zhang, X., Gu, N., Chang, J., Ye, H.: Predicting stock price movement using a dbn-rnn. Applied Artificial Intelligence **35**(12), 876–892 (2021)
42. Zhang, Y., Yan, J.: Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In: The eleventh international conference on learning representations (2022)
43. Zhao, J., Zeng, D., Liang, S., Kang, H., Liu, Q.: Prediction model for stock price trend based on recurrent neural network. Journal of Ambient Intelligence and Humanized Computing **12**, 745–753 (2021)
44. Zhou, G.B., Wu, J., Zhang, C.L., Zhou, Z.H.: Minimal gated unit for recurrent neural networks. International Journal of Automation and Computing **13**(3), 226–234 (2016)
45. Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W.: Informer: Beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 11106–11115 (2021)

A Supplementary Materials

A.1 Summary of Other Models

- **Crossformer** [42] is a transformer-based model that segments the input into a number of patches. It uses a two-stage attention layer to capture dependencies between and within variables.
- **DeformTime** [31] is a transformer-based model that includes deformable attention blocks (DABs) in addition to the encoder layers. This structure is able to capture dependencies between and within variables at different temporal granularities.
- **TSMixer** [35] is a stacked MLP based model. The stacks are interleaved - the time-domain MLPs are shared across all of the features, and the feature-domain MLPs are shared across all of the time steps.
- **PatchTST** [24] is a transformer-based model that employs segmented patches as input tokens. It predicts each variable independently, without taking into account dependencies between variables.

A.2 Hyperparameter Settings

For all the transformer based model implementations, we used default hyperparameters to generate results. The Transformer models are not pre-trained. The details of the default settings are shown below.

Training Parameters - *Crossformer* was trained with a batch size of $b = 32$ and dropout of 0.2. *DeformTime* was trained using a dropout of 0.0 and batch size of $b = 32$. *TSMixer* was trained for 200 epochs with a batch size of $b = 128$ and dropout 0.1. *PatchTST* was trained with a batch size of $b = 128$ and dropout 0.3. All the neural networks were optimised with Adam using mean squared error loss on all outputs and a starting learning rate of $1e - 04$. Training these models was set up for 100–200 training epochs, however they all reached their early stopping or pre-convergence criteria after approximately 20 epochs, as their performance was not improving.

Model parameters - All the default hyperparameter settings were used to train the respective models, The details has been compiled below for each model.

- ***Crossformer*** - The input sequence length was taken as 168 with the output windows size set to 6. The segment length was set to 6 and the window size to 2. The dimension of the hidden states in the model (embedding size) and fully connected layer was taken as $d_{model} = 256$ and $d_{fc} = 512$ respectively. The number of attention heads was 4 with 3 encoding layers.
- ***DeformTime*** - The input sequence length was taken as 336 with the prediction sequence length was 1. The initial token length was $label\_len = 0$. The patch length was taken 6 with a kernel size of 7. The encoder input and

decoder input was set to 300 and the output size to 50. The dimension of the model (embedding size) was $d_{model} = 512$ and the fully connected layer was set to $d_{fc} = 512$. The model used 4 attention heads along with 3 encoder layers and 1 decoder layer. The path length of each split was 6 with 4 strides when splitting.

- ***TSMixer*** - The input sequence length was taken as 96 with the prediction sequence length was 1. The initial token length was $label\_len = 48$. The dimension of the fully connected layers was set to $d_{fc1} = \{299, 1\}$ and $d_{fc2} = \{64, 1\}$ respectively. The number of input features in the projection layer was 299 while that in the output layer was 1.
- ***PatchTST*** - The input sequence length was taken as 256 with the prediction sequence length was 1. The encoder input was set to 6. The dimension of the model (embedding size) was set to $d_{model} = 16$ and the fully connected layer was set to $d_{fc} = 128$. The model used 4 attention heads along with 3 encoder layers. The feedforward and attention head dropouts were 0.3 and 0.0 respectively. The path length of each split was 16 with 8 strides when splitting.