NeuroFlexMLP: a Low Complexity MLP Architecture for Long-Term Time Series Forecasting

Pablo Fernández Pérez[†⋆], Claudio Fiandrino[⋆], Marco Fiore[⋆] and Joerg Widmer[⋆]
[†]Universidad Carlos III of Madrid, Madrid, Spain
[⋆]IMDEA Networks Institute, Madrid, Spain
Email: {pablo.fernandez, claudio.fiandrino, marco.fiore, joerg.widmer}@networks.imdea.org

***Abstract*—Forecasting time series over long horizons is essential for proactive decision-making in many systems. Recent research has focused on transformer-based architectures, which capture long-range dependencies in sequential data. However, several studies show that simpler linear models can outperform transformers by avoiding overfitting during training. In this context, we present NeuroFlexMLP, a deep learning model for multivariate time series forecasting tasks. NeuroFlexMLP's key distinct feature is the adaptability to the diverse complexity of real-world time series, which is achieved, from the architecture standpoint, by adding non-linear residual blocks to a first linear block. This architectural design simplifies hyperparameter optimization, leading to accurate forecasts for various time series data types regardless of the lookback or prediction horizons, outperforming state-of-the-art (SOTA) models on challenging real-world datasets. Its Multi-Layer Perceptron (MLP) design ensures high computational efficiency, making it scalable for longer input sequences than transformer-based models. We validate NeuroFlexMLP for the LEO satellite beam hopping use case, where its lightweight design enables on-board deployment, and on state-of-the art AI datasets. Across all these benchmarks, NeuroFlexMLP achieves competitive accuracy over state-of-the-art models while providing an adaptive architecture that significantly reduces computational overhead. On the LEO beam hopping task, it achieves up to 35.9% MSE reduction over Informer, which translates into up to 28% lower provisioning cost under asymmetric cost models that penalize under-allocation more heavily than over-allocation.**

I. Introduction

Time series forecasting plays a pivotal role in diverse fields, offering valuable insights into future trends and enabling informed decision-making. Applications range from finance [1] and energy management [2] to environmental sciences and climate modeling. While forecast accuracy is usually the primary evaluation criterion, computational efficiency is very important in deployment scenarios where hardware resources are scarce or energy-constrained. This is true for edge and embedded platforms that must produce forecasts in real time without access to cloud infrastructure. A prominent example is low Earth orbit (LEO) satellite communication, where on-board processors must forecast user traffic demand to drive beam hopping schedulers, yet operate under severe size, weight, and power constraints that preclude heavyweight deep learning architectures [3], [4].

Recently, the forefront of the research in time series forecasting for long-term horizon predictions has been predominantly influenced by transformer-based models such as FEDformer [5] (ICML'22), Autoformer [6] (NeurIPS'21), Informer [7] (AAAI'21 Best Paper), Pyraformer [8] (ICLR'22), among others. These models leverage multi-attention head mechanisms [9] that focus on relevant information and capture long-range dependencies, which is crucial for time series forecasting. This trend persisted until the pivotal publication of [10], showcasing significant progress achieved by the effectiveness of a single-layer linear model, which notably surpassed the SOTA transformer-based baselines. It has become apparent that the "Former family" of models encountered challenges in discerning elementary patterns within the data, even when employing complex attention-based mechanisms proper of transformer-based techniques. This deficiency was attributed to the models' susceptibility to overfitting, a drawback that becomes more pronounced when increasing the input dimensions by extending the look-back of the windows, as indicated in [10] and [11].

However, the success of linear-based models raises a compelling question: "How can a linear model achieve superior performance compared to more intricate non-linear models?" This question arises from the fact that the forecasts obtained by a non-linear model can potentially reduce to a linear solution, but not vice versa. There seems to be a distinct trade-off between identifying intricate patterns in input sequences and preventing overfitting. This delicate balance is precisely the aim that NeuroFlexMLP strives to achieve by fine-tuning the level of complexity depending on the particularities of the data in terms of sub-patterns with mixtures of trends, seasonality, and irregularities.

A compelling application domain where this complexity–accuracy tradeoff is decisive is traffic demand forecasting for beam hopping in LEO satellite networks. In beam hopping systems, the on-board scheduler dynamically allocates beam time-slots to cells based on short-term demand forecasts [3], [4]. Forecast errors translate directly into throughput loss (under-allocation) or wasted spectrum (over-allocation). Prior work applied the Informer Transformer [7] to this task, only marginally outperforming classical FARIMA while incurring substantial computational overhead, which is a critical limitation given the constrained processing capabilities of LEO satellites. NeuroFlexMLP addresses precisely this gap, offering a lightweight architecture that captures multi-scale temporal patterns at a fraction of the computational cost.

The architecture of NeuroFlexMLP revolves around organizing the model into a chain of residual blocks preceded by a linear block. The non-linear blocks consist of two linear layers separated by an activation function, while the initial block is entirely linear. NeuroFlexMLP adapts to the data characteristics by increasing its complexity via the manipulation of only two hyperparameters: the number of non-linear layer blocks and the block's hidden size.

The minimal architectural design of NeuroFlexMLP makes it highly efficient in comparison to transformer-based models. Efficient machine learning models are crucial to meet the requirements of today's real-world datasets, whose volumes of data continue to grow exponentially. The adoption of inefficient models can pose significant challenges, particularly

in scenarios where computational resources are limited or costly. Therefore, developing and deploying efficient models is essential not only for optimizing performance but also for minimizing computational overheads and promoting broader adoption across diverse domains and applications. The MLP design of NeuroFlexMLP enables the parallelization of multiple operations without a substantial increase in training times. Thus, NeuroFlexMLP can process longer input sequences without increasing computational overhead, which improves its capacity to model long-range dependencies in the data.

The synopsis of the contribution of this work is as follows.

- We propose NeuroFlexMLP, an MLP-based model for long-term horizon forecasting tasks that is specifically designed to trade architecture complexity and forecast accuracy.
- To achieve the above tradeoff, our model incorporates only two hyperparameters and is constructed as a series of residual non-linear blocks preceded by a linear layer. These residual blocks facilitate the learning of non-linearities in the data only when necessary.
- We validate NeuroFlexMLP on a LEO satellite beam hopping traffic forecasting use case, demonstrating up to 35.9% MSE improvement over Informer with a lower computational cost, and confirming through scheduler simulation that NeuroFlexMLP yields up to 28% lower total provisioning cost under an asymmetric cost framework inspired by [12].
- We perform a comprehensive set of empirical studies to benchmark NeuroFlexMLP against various baselines, including transformer- and MLP-based models. Our studies delve into the capability of modeling long inputs, computational complexity analysis, and explainability.

Our findings show that NeuroFlexMLP achieves comparable accuracy performance to transformers but with significantly lower overall complexity. While transformer-based baselines can incur an exponential learning cost, NeuroFlexMLP's learning cost is almost invariant to the look-back size. Its architectural design introduces minimal complexity (i.e., two main hyperparameters) compared to linear-based baselines, making efficient the hyperparameter search while improving significantly the prediction accuracy.

II. Related Work

Transformers-based models. Transformers excel at capturing long-range dependencies in time series forecasting. Popular architectures include LogTrans [13], which uses convolutional self-attention and a LogSparse mechanism to improve local context and reduce memory overhead. Autoformer [6] and FEDformer [5] introduce deep decomposition schemes coupled with series-wise auto-correlation and frequency-enhanced attention, respectively. Pyraformer [8] uses a pyramid structure to efficiently capture hierarchical temporal dependencies. More recently, PatchTST [11] mitigates overfitting and enhances locality through patching and channel independence. Patching enhances locality and provides semantic meaning to input tokens, while channel independence addresses overfitting of previous transformer models. Finally, iTransformer [14] inverts the traditional structure of transformers to embed entire series instead of individual time steps.

MLP-based models. MLP-based architectures have gained traction for their simplicity and efficiency. TSMixer [15] extracts features by mixing time and feature dimensions across stacked MLPs. NBEATS [16] uses double residual connections for explainable trend and seasonality learning, while NHITS [17] uses multi-rate sampling and hierarchical interpolation to reduce computational complexity. DLinear [10] demonstrates that a straightforward linear model with a decomposition scheme can achieve remarkable accuracy at a fraction of the cost of transformers. Recently, PatchMLP [18] integrated patching into pure MLPs to capture local temporal information. However, unlike these approaches, NeuroFlexMLP remains unique in its use of statistical nonlinearity tests to dynamically modulate architectural depth.

Forecasting in LEO satellite networks. The rapid deployment of LEO mega-constellations such as Starlink has attracted both measurement and systems research. Early efforts revealed the performance landscape of commercial LEO services: Raman et al. [19] compared multiple operators, while Li et al. [20] provided the first networking-centric analysis of Starlink's autonomous orbital management. Liu et al. [21] dissected Starlink throughput variability and proposed StarNet, a learning-based predictor incorporating satellite-specific features. The unique dynamics of LEO links such as frequent handovers, rapidly changing topologies, and time-varying capacity have motivated new protocol designs: Lai et al. [22] redesigned congestion control for periodic satellite reconfigurations, Tao et al. [23] proposed Umbra, a scheduler exploiting predictable orbital contact windows, and Li et al. [24] showed how small-scale constellations can serve global demand through intelligent resource allocation. Complementary measurement platforms include the work by Izhikevich et al. [25] and StarryNet by Lai et al. [26]. These works identify that prediction is relevant yet not well explored: Casparsen et al. [27] showed that Starlink end-to-end latency follows a deterministic 15-second periodic pattern, while Demirci et al. [3], [4] applied the Informer Transformer to forecast self-similar user traffic for beam hopping.

III. NeuroFlexMLP

A. Problem formulation

Let $X \in \mathbb{R}^{C \times L}$ represent the C historical sequences (variates) of length L, and let $Y \in \mathbb{R}^{C \times T}$ denote the C sequences of future values of length T. The primary objective is to identify a function that, when given X as input, produces a prediction $\hat{Y}$ that closely approximates the ground truth Y.

We adopt the common methodology of partitioning the original time series data $\mathcal{X} \in \mathbb{R}^{C \times S}$ of size S into multiple subsequences $X \in \mathbb{R}^{C \times L}$, ensuring that neighboring subsequences overlap.

B. Architecture

Overall design. The design of NeuroFlexMLP is based on a block-based architecture, with each block consisting of two linear layers with an activation function in between, except for the initial block, which is fully composed of linear layers. The block design and the overall architecture are illustrated in Fig. 1. The blocks are concatenated, implying that the input of a block is the output of its predecessor. Additionally, in each block, except for the first one, a residual connection is incorporated, meaning that the block's output is the summation of the block input sequence and the last linear layer output. This design allows for the creation of a highly flexible model, streamlining the hyperparameter search to just two main hyperparameters: the number of non-linear blocks and their hidden size. These two hyperparameters enable the model

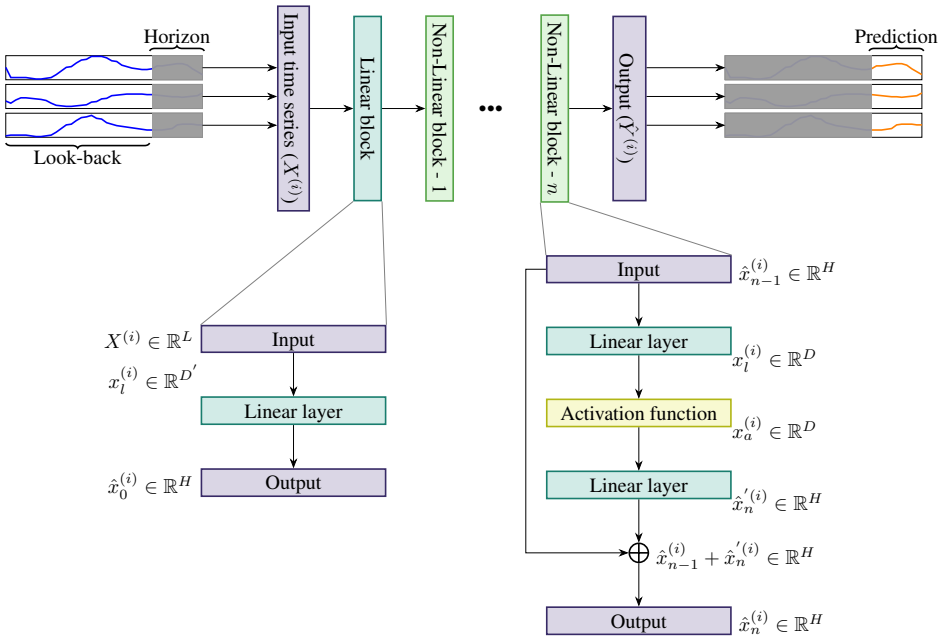


Fig. 1. Architectural design of NeuroFlexMLP. The architecture of NeuroFlexMLP is designed to be channel-independent. The input sequence X from variate i and length L is fed into a first linear block with hidden size D', which generates an output aligned with the desired prediction dimensions H. Subsequently, there is a chain of nonlinear blocks with a similar structure to the linear block but include a ReLU activation function between the linear layers, adding non-linearity to the model. The input of each non-linear block, denoted as $\hat{x}_{n-1}$, is the output of the previous block. Additionally, there is a residual connection that sums the block's input with the output of its last linear layer, denoted as $\hat{x}^{'(i)}$.

to adapt its complexity to various types of time series data. Each hyperparameter contributes differently to the model. Increasing the number of blocks enables the discovery of more intricate patterns at the expense of incurring overfitting, while the hidden size determines the volume of operations performed in each block.

First linear block. The design choice of including an initial linear block is motivated by its capacity to lay a stable foundation for any subsequent non-linear transformation, providing an underfitted but reasonably accurate starting point for the next blocks to refine the final predictions. This design enables us to obtain remarkable results while maintaining the model structure as simple as possible.

Non-linear blocks. The chain of non-linear blocks is designed to incrementally add non-linearity to the output of the first linear block. We have opted for this chain-based design to retain simplicity in the block structure while allowing significant complexity variations when manipulating the hidden size. We experimented with alternative block designs, such as incorporating uni-dimensional convolutional layers or recurrent layers like Long Short-Term Memory (LSTM) [28]. However, these designs were ultimately discarded as they did not yield accuracy improvements while introducing unnecessary complexity.

To determine the optimal number of non-linear residual blocks, we propose a data-driven decision procedure that operates entirely on the training set prior to model fitting (see Fig. 2). The procedure first applies two data gates: if the ratio of the number of available samples to input channels is lower than 10, or if the time series is univariate ($n_{\text{channels}} = 1$), the model does not use any non-linear blocks since scarce data cannot support additional model depth without overfitting, and univariate series benefit more from a wide input window than from stacked non-linearities. Otherwise, three different statistics are computed on the training data: *(i)* the Brock-Dechert-Scheinkman (BDS) test statistic [29], applied to the residuals of a fitted autoregressive AR(p) model selected via the Akaike Information Criterion (AIC), which quantifies the degree of non-linear serial dependence that persists after removing linear dynamics; *(ii)* the seasonal strength S, defined as the fraction of spectral power concentrated at the dominant frequency of the detrended series; and *(iii)* the trend strength T, defined as the fraction of variance explained by a linear trend (R^2 of a least-squares fit). The BDS statistic drives a sequence of threshold decisions. If BDS < 5, linear dynamics are more important than non-linear ones, thus the architecture features zero non-linear layers. If $5 \leq$ BDS < 8, the time series exhibits mild non-linearity: one block is assigned by default, unless seasonal strength exceeds 0.30, in which case two blocks are used to capture the periodic component. If $8 \leq$ BDS < 12, sub-conditions on seasonality and trend further refine the number of non-linear layers: jointly strong seasonality ($S > 0.35$) and trend ($T > 0.15$) yield four blocks for complex regime dynamics, seasonality alone ($S > 0.30$) yields three, and the absence of both yields two non-linear blocks. Finally, if BDS ≥ 12, which indicates strong non-linear dynamics, three blocks are assigned. This decision scheme was validated in 11 well-established datasets, including ETT [7], Weather, Electricity, Traffic [6], and National Illness. The procedure yields an exact match with the empirically optimal layer count in 10 out of 11 cases (MAE $= 0.09$), with a single mismatch (ETTm2, BDS $= 3.9$, below the threshold of 5) that falls within the ± 1 layer.

While the BDS test involves a correlation integral calculation, it is performed as a one-time offline pre-processing step prior to model fitting. Its overhead is $O(S \log S)$, which is negligible compared to the iterative backpropagation costs of Transformer-based alternatives. Furthermore, this procedure provides a form of structural explainability: practitioners can inspect the BDS statistic, seasonal strength, and trend strength of their data to understand *why* the model adopts a given depth, in contrast to post-hoc explainability approaches that

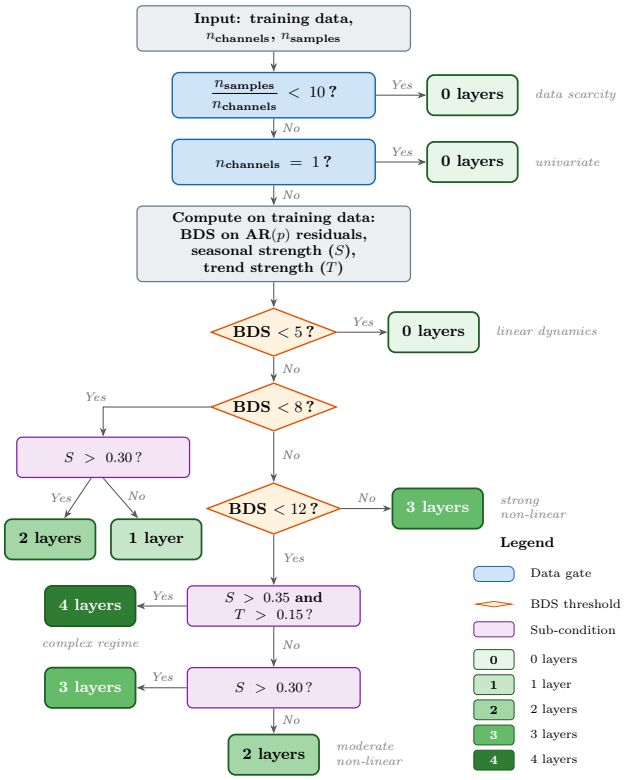


Fig. 2. Methodology to select the number of non-linear blocks

analyze a trained model's internal representations [30], [31].

Residual connections. Residual connections [32] play a crucial role in enabling the construction of deeper models. They facilitate training convergence by mitigating the vanishing gradient problem, thereby enabling the successful training of neural networks with a greater number of layers. Besides this benefit, the addition operation within the residual connection results in the output being only a minor variation of the input, thus significantly mitigating overfitting. In contrast, alternative design options like concatenation would over-complicate the model complexity while not yielding accuracy improvements.

Channel Independence. The model is designed to be channel-independent, which means that operations do not involve cross-channel interactions. For this design choice, we take inspiration from [11] to mitigate the risk of overfitting during the training process. We noticed notable improvements in model accuracy when including this feature in the design. By prioritizing channel independence, NeuroFlexMLP generalizes well across diverse datasets and scenarios. This approach also simplifies the learning process, allowing the model to focus on capturing relevant patterns within individual channels without being overly influenced by inter-channel correlations.

Normalization. Each input sequence undergoes a zero-mean normalization process through the application of a Reversible Instance Normalization (RevIn) layer [33]. The effectiveness of this normalization process has been well-documented, demonstrating a notable improvement in model accuracy. Similar normalization techniques have been implemented in SoTA models such as PatchTST [11] and TS-Mixer [15].

Decomposing Configuration. NeuroFlexMLP includes an optional configuration designed for decomposing time series data. This configuration is responsible for transforming the input sequence into distinct trend and seasonal components at the beginning of the forward function, drawing inspiration from methodologies presented in DLinear [10] and Autoformer [6]. When decomposing, the non-linear blocks' inputs consist of the trend $x_{tr}^{(i)}$ and seasonal component $x_{se}^{(i)}$ in addition to the original block input $\hat{x}_{n-1}^{(i)}$. First linear block input and operations remain unaltered. The block's output is $\hat{x}_n^{(i)} = \hat{x}_{n-1}^{(i)} + \hat{x}_n^{\prime(i)} + \hat{x}_{tr}^{\prime(i)} + \hat{x}_{se}^{\prime(i)}$. It is important to note that each $\hat{x}_n^{\prime(i)}, \hat{x}_{tr}^{\prime(i)}, \hat{x}_{se}^{\prime(i)}$ is passed through individual linear layers, ensuring that the block's layer parameters are not shared. Furthermore, it should be emphasized that the decomposed components $x_{tr}^{(i)}$, $x_{se}^{(i)}$ used as inputs remain constant throughout the different blocks and correspond to the decomposed input sequence X, in contrast to $\hat{x}_{n-1}^{(i)}$ that is the output of the preceding block. See Fig. 1 for a more clear view of the notation.

IV. LEO Beam Hopping Traffic Forecasting

A. Methodology

Dataset. We evaluate NeuroFlexMLP on a LEO satellite beam hopping traffic forecasting task, reproducing the synthetic dataset methodology of Demirci et al. [3]. The dataset models aggregate user traffic demand as the superposition of M independent ON/OFF sources whose active and idle durations follow heavy-tailed Pareto distributions with shape parameter a. This construction yields self-similar aggregate traffic with Hurst exponent $H = (3 - a)/2$, capturing the empirically observed long-range dependence in real satellite user traffic [4]. We consider three demand scenarios: *high* (a=1.04, H=0.98, M=750), *medium* (a=1.6, H=0.70, M=500), and *low* (a=1.9, H=0.55, M=250), where each terminal contributes 1 Mbps while active. Traffic is generated at 10 ms base granularity and aggregated to 1 s slots, yielding 60000 samples per scenario. The target variable is the aggregate traffic demand in Mbps.

The Use Case. The LEO beam hopping problem consists of allocating dynamically beam time-slots to cells based on demand forecasts. Under-prediction causes buffer overflow and packet drops; over-prediction wastes beam resources that are precious because they are scarce. We compare NeuroFlexMLP with Informer [7], the forecasting model used in [3], for prediction horizons $T \in \{1, 12, 24, 48\}$ and lookback windows $L \in \{128, 512\}$. Beyond standard forecast accuracy (MSE, MAE), we assess the impact of forecasts on the network through a finite-buffer scheduler simulation: a queue per-cell is driven by forecast-based capacity allocation (capacity = predicted demand $\times$ (1 + margin%)), and we measure throughput loss, drop rate, and capacity waste across overprovision margins of 0% to 50%.

B. Results

Forecasting accuracy. Figure 3 shows that NeuroFlexMLP outperforms Informer in the high-demand scenario (H=0.98) in all 8 configurations (4 prediction horizons T and 2 lookback windows L) with an average MSE improvement of 12.7% over Informer and a peak 35.9% for L=512, T=1. In the medium scenario (H=0.70), both models perform comparably. In the low scenario (H=0.55), the Hurst exponent is only marginally above the white-noise baseline of H=0.5, leaving minimal temporal structure for either model to exploit. This explains why MSE for both models approaches 1.0. Overall, the results confirm that NeuroFlexMLP's linear backbone is effective at capturing the strong long-range structure present in high-Hurst traffic, precisely the regime that is most relevant for practical beam hopping scheduling.

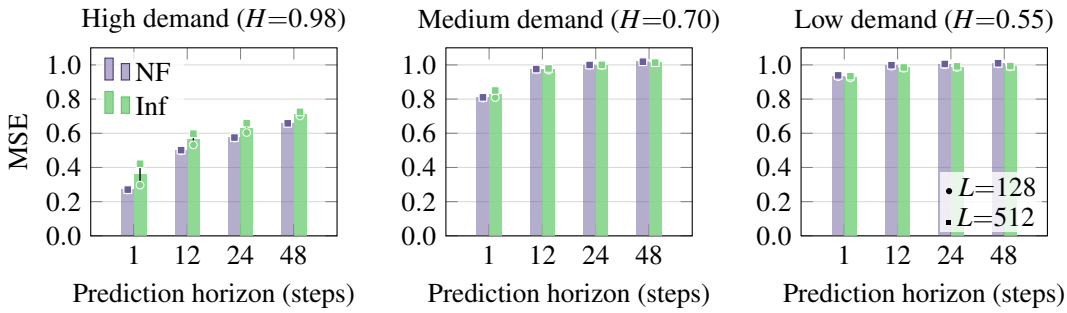


Fig. 3. Forecast accuracy of NeuroFlexMLP (NF) and Informer (Inf) on LEO beam hopping problem. NF works especially well in critical high-demand scenarios ($H = 0.98$), which demonstrates that its lightweight design successfully captures the essential long-range dependencies of satellite traffic.

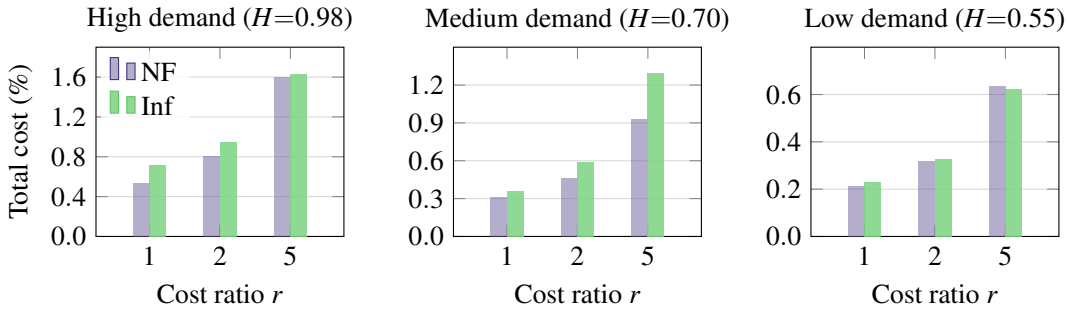


Fig. 4. Total provisioning cost of NeuroFlexMLP (NF) and Informer (Inf) under different cost ratios. NeuroFlexMLP translates its predictive advantage into up to 28% lower total provisioning costs.

Networking-level impact. Inspired by [12], we evaluate both models under a provisioning cost function $C = \alpha \cdot C_{\text{under}} + \beta \cdot C_{\text{over}}$, where C_{under} and C_{over} denote throughput loss and capacity waste, respectively, and the cost ratio $r = \alpha/\beta \geq 1$ captures the penalty of under-provisioning. In beam hopping systems, under-provisioning a cell means insufficient beam time-slots are allocated to meet user demand: excess traffic fills the on-board buffer and, once full, packets get dropped, causing service degradation and potential SLA violations that cannot be recovered in the next scheduling period. In LEO, retransmission may have to wait for the next scheduling window or even the next satellite pass. Excessive over-provisioning, by contrast, can waste beam resources that could serve other cells. Figure 4 reports the total cost when varying the ratio r. We choose $r \in \{1, 2, 5\}$. To illustrate, $r = 5$ means that the operator considers losing 1 unit of throughput to be $5\times$ more costly than wasting 1 unit of beam capacity while the setting $r = 1$ treats capacity waste and loss equally. NeuroFlexMLP achieves a lower cost than Informer in all cost ratios in both the high-demand (H=0.98) and medium-demand (H=0.70) scenarios, with the advantage growing as under-provisioning becomes costlier: at r=5, NeuroFlexMLP reduces the total provisioning cost by 28% in the medium scenario. In the low-demand scenario (H=0.55), both models perform comparably, with differences below 2%. These results confirm that NeuroFlexMLP's lower forecast error translates into effective resource allocation specifically in the high-variability regimes that are the most relevant for real-world beam hopping scheduling.

V. Validating NeuroFlexMLP in AI Benchmarks

A. Long-Horizon Time Series Forecasting Results

Datasets. We evaluate our proposed model on eight datasets commonly used as a benchmark for comparing long horizon time series forecasting models and provided in [6]. The datasets are `Weather`, `Traffic`, `Electricity`, `ILI`, `ETTh1`, `ETTh2`, `ETTm1`, `ETTm2`. Table I lists the features of each dataset, variates, and size. Specifically, `Weather` contains 21 meteorological indicators for a range of one year in Germany. `Traffic` contains the road occupation in 862 different locations of San Francisco Bay. `Electricity` dataset measures the electricity consumption of 321 customers. `ILI` contains the recorded USA influenza-like illness patients. `ETT` datasets include some indicators such as load and oil temperature from an electricity transformer with hourly (`ETTh`) and fifteen-minute granularity (`ETTm`).

We set the prediction length to $T \in \{96, 192, 336, 720\}$ for the datasets except for `ILI` where we set the prediction length to $T \in \{24, 36, 48, 60\}$ because of the smaller size of the dataset. The look-back window L varies across datasets and horizons, and it is defined as $L = \alpha \cdot T$, with $1 \leq \alpha \leq 6$.

Baselines. We select the latest SOTA models as baselines for accuracy comparison (i.e., MAE and MSE) with NeuroFlexMLP. Specifically, we benchmark against models of the transformer-based family and MLP-based ones. Our objective is to compare NeuroFlexMLP against models yielding high accuracy at the expense of high model complexity (i.e., transformer-based models), and against models that exhibit a cost-effective architectural design like ours.

Concerning the transformer-based models, we benchmark against recent architectures like PatchTST [11], Pyraformer [8], and FedFormer [5]. Additionally, we include first-generation transformer baselines, *i.e.*, Autoformer [6], Informer [7], and LogTrans [13] to contextualize our performance and demonstrate the efficiency gains of modern low-complexity approaches. For the MLP-based models, we use TSmixer [15] and DLinear [10].

Model Accuracy. We evaluated the accuracy of NeuroFlexMLP in both multivariate and univariate tasks and compared it to the baselines. We assess the performance of the

TABLE I
DATASETS FEATURES.

Datasets	Weather	Traffic	Electricity	ILI	ETTh1	ETTh2	ETTm1	ETTm2
Variates	21	862	321	7	7	7	7	7
Granularity	10 minutes	Hourly	Hourly	Weekly	Hourly	Hourly	15 minutes	15 minutes
Timesteps	52696	17544	26304	966	17420	17420	69680	69680

TABLE II
PERFORMANCE COMPARISON FOR MULTIVARIATE LONG-HORIZON FORECASTING. THE TABLE HIGHLIGHTS THE BEST RESULTS IN BOLD AND THE SECOND-BEST RESULTS ARE UNDERLINED. THE FORECAST HORIZONS ARE DENOTED AS $T = \{96, 192, 336, 720\}$ FOR ALL DATASETS EXCEPT FOR ILI, WHERE $T = \{24, 36, 48, 60\}$. THE "-" SYMBOL INDICATES THAT RESULTS WERE NOT PROVIDED BY THE ORIGINAL AUTHORS. NEUROFLEXMLP RESULTS ARE THE MEAN OF FIVE RUNS WITH DIFFERENT RANDOM SEEDS. THE PATCHTST RESULTS REPRESENT THE BEST MSE SCORE BETWEEN THE TWO VERSIONS OF THE MODEL, NAMELY PATCHTST/64 AND PATCHTST/48 WHICH WERE BOTH PROVIDED BY THE ORIGINAL AUTHORS. THE REMAINING TRANSFORMER-BASED RESULTS ARE SOURCED FROM [11].

Models		NeuroFlexMLP		PatchTST		TS-Mixer		DLinear		FEDFormer		Autoformer		Informer		Pyraformer		LogTrans	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Weather	96	0.146	**0.196**	0.149	0.198	**0.145**	0.198	0.176	0.237	0.238	0.314	0.249	0.329	0.354	0.405	0.896	0.556	0.458	0.490
	192	**0.191**	**0.241**	0.194	**0.241**	**0.191**	0.242	0.220	0.282	0.275	0.329	0.325	0.370	0.419	0.434	0.622	0.624	0.658	0.589
	336	0.244	0.285	0.245	0.282	**0.242**	**0.280**	0.265	0.319	0.339	0.377	0.351	0.391	0.583	0.543	0.739	0.753	0.797	0.652
	720	**0.297**	**0.330**	0.314	0.334	0.320	0.336	0.323	0.362	0.389	0.409	0.415	0.426	0.916	0.705	1.004	0.934	0.869	0.675
Traffic	96	0.366	0.259	**0.360**	**0.249**	0.376	0.264	0.410	0.282	0.576	0.359	0.597	0.371	0.733	0.410	2.085	0.468	0.684	0.384
	192	**0.368**	**0.256**	0.379	**0.256**	0.397	0.277	0.423	0.287	0.610	0.380	0.607	0.382	0.777	0.435	0.867	0.467	0.685	0.390
	336	**0.384**	0.273	0.392	**0.264**	0.413	0.290	0.436	0.296	0.608	0.375	0.623	0.387	0.776	0.434	0.869	0.469	0.734	0.408
	720	**0.428**	0.291	0.432	**0.286**	0.444	0.306	0.466	0.315	0.621	0.375	0.639	0.395	0.827	0.466	0.881	0.473	0.717	0.396
Electricity	96	**0.129**	0.224	**0.129**	**0.222**	0.131	0.229	0.140	0.237	0.186	0.302	0.196	0.313	0.304	0.393	0.386	0.449	0.258	0.357
	192	**0.139**	**0.235**	0.147	0.240	0.151	0.246	0.153	0.249	0.197	0.311	0.211	0.324	0.327	0.417	0.386	0.443	0.266	0.368
	336	**0.155**	**0.252**	0.163	0.259	0.161	0.261	0.169	0.267	0.213	0.328	0.214	0.327	0.333	0.422	0.378	0.443	0.280	0.380
	720	**0.182**	**0.282**	0.197	0.290	0.197	0.293	0.203	0.301	0.233	0.344	0.236	0.342	0.351	0.427	0.376	0.445	0.283	0.376
ILI	24	1.433	0.762	**1.319**	**0.754**	-	-	2.215	1.081	2.624	1.095	2.906	1.182	4.657	1.449	1.420	2.012	4.480	1.444
	36	**1.379**	**0.787**	1.430	0.834	-	-	1.963	0.963	2.516	1.021	2.585	1.038	4.650	1.463	7.394	2.031	4.799	1.467
	48	1.750	0.866	**1.553**	**0.815**	-	-	2.130	1.024	2.505	1.041	3.024	1.145	5.004	1.542	7.551	2.057	4.800	1.468
	60	1.684	0.855	**1.470**	**0.788**	-	-	2.368	1.096	2.742	1.122	2.761	1.114	5.071	1.543	7.662	2.100	5.278	1.560
ETTh1	96	0.367	0.394	0.370	0.400	**0.361**	**0.392**	0.375	0.399	0.376	0.415	0.435	0.446	0.941	0.769	0.664	0.612	0.878	0.740
	192	0.409	0.424	0.413	0.429	**0.404**	0.418	0.405	**0.416**	0.423	0.446	0.456	0.457	1.007	0.786	0.790	0.681	1.037	0.824
	336	0.469	0.481	0.422	0.440	**0.420**	**0.431**	0.439	0.443	0.444	0.462	0.486	0.487	1.038	0.784	0.891	0.738	1.238	0.932
	720	0.468	0.477	**0.447**	**0.468**	0.463	0.472	0.472	0.490	0.469	0.492	0.515	0.517	1.144	0.857	0.963	0.782	1.135	0.852
ETTh2	96	**0.269**	**0.334**	0.274	0.337	0.274	0.341	0.289	0.353	0.332	0.374	0.332	0.368	1.549	0.952	0.645	0.597	2.116	1.197
	192	**0.327**	**0.375**	0.339	0.379	0.339	0.385	0.383	0.418	0.407	0.446	0.426	0.434	3.792	1.542	0.788	0.683	4.315	1.635
	336	0.343	0.405	**0.329**	**0.384**	0.361	0.406	0.448	0.465	0.400	0.447	0.477	0.479	4.215	1.642	0.907	0.747	1.124	1.604
	720	0.400	0.442	**0.379**	**0.422**	0.445	0.470	0.605	0.551	0.412	0.469	0.453	0.490	3.656	1.619	0.963	0.783	3.188	1.540
ETTm1	96	**0.285**	0.340	0.290	0.342	**0.285**	**0.339**	0.299	0.343	0.326	0.390	0.510	0.492	0.626	0.560	0.543	0.510	0.600	0.546
	192	0.339	0.371	0.332	0.369	**0.327**	**0.365**	0.335	**0.365**	0.365	0.415	0.514	0.495	0.725	0.619	0.557	0.537	0.837	0.700
	336	0.358	0.387	0.366	0.392	**0.356**	**0.382**	0.369	0.386	0.392	0.425	0.510	0.492	1.005	0.741	0.754	0.655	1.124	0.832
	720	**0.407**	0.418	0.416	0.420	0.419	**0.414**	0.425	0.421	0.446	0.458	0.527	0.493	1.133	0.845	0.908	0.724	1.153	0.820
ETTm2	96	0.166	0.255	0.165	0.255	**0.163**	**0.252**	0.167	0.260	0.180	0.271	0.205	0.293	0.355	0.462	0.435	0.507	0.768	0.642
	192	**0.215**	0.293	0.220	0.292	**0.216**	**0.290**	0.224	0.303	0.252	0.318	0.278	0.336	0.595	0.586	0.730	0.673	0.989	0.757
	336	**0.260**	0.327	0.274	0.329	0.268	**0.324**	0.281	0.342	0.324	0.364	0.343	0.379	1.270	0.871	1.201	0.845	1.334	0.872
	720	**0.327**	**0.373**	0.362	0.385	0.420	0.422	0.397	0.421	0.410	0.420	0.414	0.419	3.001	1.267	3.625	1.451	3.048	1.328

multivariate setting across the entire benchmark dataset, while for the univariate, we focus on the `ETT` dataset, where the target variable to predict is `Oil Temperature`. Table II displays the results for the multivariate setting, while Table III presents the results for the univariate setting. NeuroFlexMLP outperforms PatchTST in the majority of datasets, except `ETTh` and `ILI`. In the latter case, we attribute this discrepancy in performance to the limited extension of the `ILI` dataset, which makes hard the learning process during training. To verify whether our intuition was correct, we retrained both NeuroFlexMLP and PatchTST on an extended version of the ILI dataset that includes data from 2020 through January 2024, obtained from the CDC FluView portal. On this extended dataset, the performance gap narrows substantially and NeuroFlexMLP wins three out of four horizons (e.g., MSE=5.23 vs. 7.39 at T=24), confirming that the original `ILI` disadvantage stems from the limited dataset size rather than a model limitation. Furthermore, NeuroFlexMLP demonstrates higher performance for longer horizons, particularly excelling at horizon 720 in 5 out of the 8 datasets, and for `Traffic`, `Electricity`, and `Weather` datasets, demonstrating its ability to predict time series with a high number of variates.

B. Implementation

Training. We iterate over 100 epochs for each dataset to train NeuroFlexMLP. Throughout the training process, we utilized the Mean Squared Error (MSE) loss function (see Eq. 1) to quantify the disparity between predicted and actual values. For optimization, we used the `ADAM` optimizer [34]. Additionally, a `ReduceLrOnPlateau` scheduler was utilized to dynamically reduce the learning rate when the MSE stopped improving.

$$\text{MSE} = \frac{1}{C \cdot W} \sum_{c=1}^{C} \sum_{w=1}^{W} (Y_{c,w} - \hat{Y}_{c,w})^2, \tag{1}$$

where $c = 1, \ldots, C$ represents the variates, $w = 1, \ldots, W$ denotes the windows, and Y and $\hat{Y}$ denote the ground truth and the predictions, respectively.

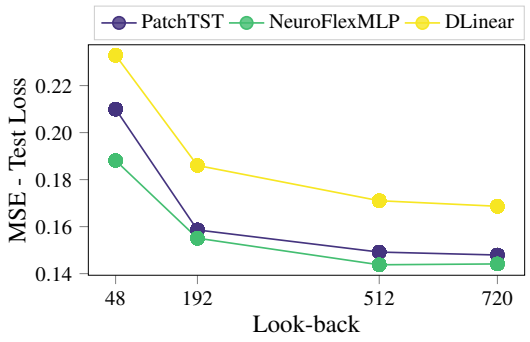
Hyperparameter search. In our experiments, we employed `optuna`, a hyperparameter optimization framework [35]. `optuna` makes it possible to systematically conduct the hyperparameter search, delving into the number of non-linear blocks, and the dimensions of the non-linear block's hidden size. `optuna` also allows us to experiment with other key parameters such as the sequence length.

C. Hyperparameter Analysis

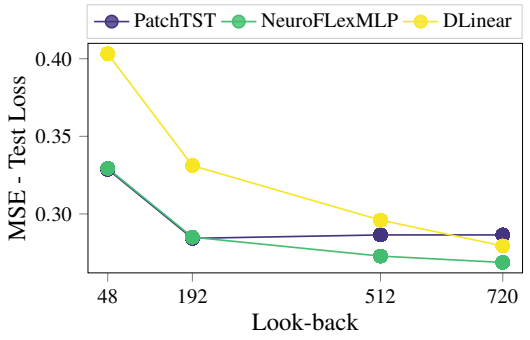
Varying Input Sequence Length. In principle, the larger size of the input sequence should result in performance enhancements, as models gain access to a greater amount of information to learn from and make predictions. However,

TABLE III
RESULTS FOR UNIVARIATE LONG-TERM FORECASTING. HORIZONS ARE $T \in \{96, 192, 336, 720\}$. THE BEST RESULTS ARE HIGHLIGHTED IN **BOLD**. TS-MIXER UNIVARIATE RESULTS WERE NOT PROVIDED BY THE AUTHORS.

Models		NeuroFlexMLP		PatchTST		DLinear		FEDFormer		Autoformer		Informer		LogTrans	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.058	0.187	**0.055**	**0.179**	0.056	0.180	0.079	0.215	0.071	0.206	0.193	0.377	0.283	0.468
	192	0.077	0.218	**0.071**	0.205	**0.071**	**0.204**	0.104	0.245	0.114	0.262	0.217	0.395	0.234	0.409
	336	0.083	0.229	**0.076**	**0.220**	0.098	0.244	0.119	0.270	0.107	0.258	0.202	0.381	0.386	0.546
	720	0.093	0.244	**0.087**	**0.236**	0.189	0.359	0.142	0.299	0.126	0.283	0.183	0.355	0.475	0.629
ETTh2	96	0.132	0.284	**0.129**	0.282	0.131	**0.279**	0.128	0.271	0.153	0.306	0.213	0.373	0.217	0.379
	192	0.180	0.337	**0.168**	**0.328**	0.176	0.329	0.185	0.330	0.204	0.351	0.227	0.387	0.281	0.429
	336	0.186	0.349	**0.171**	**0.336**	0.209	0.367	0.231	0.378	0.246	0.389	0.242	0.401	0.293	0.437
	720	0.241	0.392	**0.223**	**0.380**	0.276	0.426	0.278	0.420	0.268	0.409	0.291	0.439	0.218	0.387
ETTm1	96	**0.025**	**0.121**	0.026	0.123	0.028	0.123	0.033	0.140	0.056	0.183	0.109	0.277	0.049	0.171
	192	**0.038**	**0.149**	0.039	0.150	0.045	0.156	0.058	0.186	0.081	0.216	0.151	0.310	0.157	0.317
	336	**0.050**	**0.171**	0.053	0.174	0.061	0.182	0.084	0.231	0.076	0.218	0.427	0.591	0.289	0.459
	720	**0.068**	**0.200**	0.073	0.206	0.080	0.210	0.102	0.250	0.110	0.267	0.438	0.586	0.430	0.579
ETTm2	96	0.065	0.189	0.065	0.187	**0.063**	**0.183**	0.067	0.198	0.065	0.189	0.088	0.225	0.075	0.208
	192	0.094	0.233	0.093	0.231	**0.092**	**0.227**	0.102	0.245	0.118	0.256	0.132	0.283	0.129	0.275
	336	0.126	0.274	0.120	0.265	**0.119**	**0.261**	0.130	0.279	0.154	0.305	0.180	0.336	0.154	0.302
	720	**0.168**	0.321	0.172	0.322	0.175	**0.320**	0.178	0.325	0.182	0.335	0.300	0.435	0.160	0.321



(a) `Weather`



(b) `ETTm2`

Fig. 5. Impact of the look-back window on the model accuracy for `Weather` and `ETTm2` datasets.

as demonstrated in [11] and [10], there are cases where this is not necessarily true, especially when models overfit. NeuroFlexMLP effectively captures relevant information with increased look-back window sizes, as illustrated in Fig. 5. We also compare it with PatchTST and DLinear, two models known to benefit from larger input sequences, demonstrating similar or even superior performance gains.

Number of blocks and Hidden size. The number of blocks and the hidden size within each block are two parameters that NeuroFlexMLP can adjust to incorporate non-linear operations. We kept one of these hyperparameters fixed while varying the other to observe its impact on the model's performance. We illustrate the results in Fig. 6 for Weather (6a, 6c) and ILI datasets (6b, 6d). It can be seen that there is a clear optimal point for both hyperparameters in terms of MSE. This suggests that there is only one best configuration for a given choice of one hyperparameter.

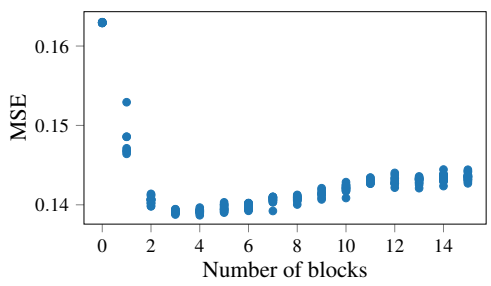
D. Computational Complexity

We measured CUDA training and inference times and the accuracy of NeuroFlexMLP, PatchTST, DLinear, TSMixer and Crossformer. We trained the models for input sequences $L \in \{256, 512, 1024\}$ and horizons $T \in \{96, 192, 336, 720\}$ for all the datasets, but `ILI` and $L \in \{48, 64, 82, 104\}$ and $T \in \{24, 36, 48, 60\}$ for `ILI` dataset. We specifically measured the CUDA training times per epoch using the Pytorch profiler to isolate the duration of the operations executed on the GPU.

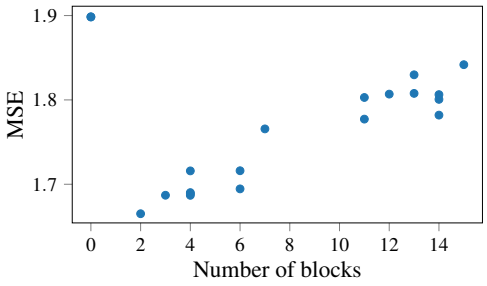
We sequentially executed all models on the same machine, *i.e.*, a Dell PowerEdge XE8545 server equipped with 4 Nvidia A100 - 40 GB. Each model was implemented using the PyTorch framework. The batch size was fixed for all the models, while the remaining hyperparameters were tuned to their optimal values for each dataset.

In our experiments, we observed a clear trade-off between accuracy and complexity for NeuroFlexMLP compared to the other models, as illustrated in Fig. 7. Additionally, we found that our model excels in CUDA training times per epoch, achieving values very close to DLinear, which is an extremely simple model with just one linear layer. We also noted that transformer-based models, such as PatchTST and CrossFormer, quadratically increase in complexity with larger lookback windows. This exponential increase makes it challenging to use longer input sequences, which can be crucial for datasets with long-range patterns.

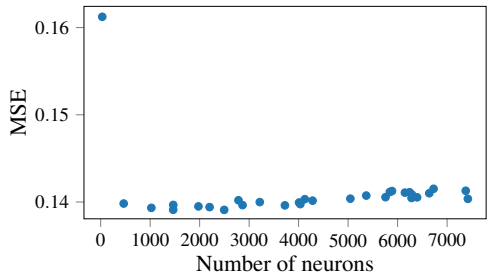
Figure 8 reports CUDA training and inference times on the Electricity dataset. NeuroFlexMLP's per-epoch training cost remains nearly constant as L grows from 256 to 1024, close to DLinear. In contrast, transformer-based models (PatchTST, CrossFormer) scale quadratically: NeuroFlexMLP is $6\times$ and $17\times$ faster in training and inference than PatchTST for $L = 256$ and $L = 512$ respectively while at L=1024 PatchTST exceeds GPU memory. This gap holds true on the beam hopping dataset, where Informer requires $\sim$49 s/epoch at L=512 versus $\sim$0.5 s/epoch for NeuroFlexMLP, which is a $\sim 98\times$ speedup. Even at the shorter lookback L=128, NeuroFlexMLP trains $\sim 43\times$ faster per epoch.



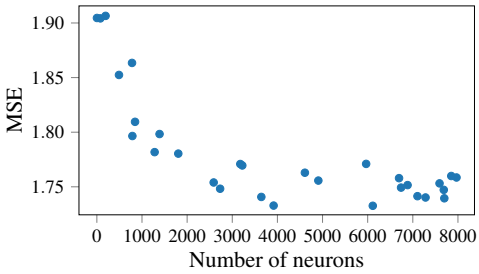
(a) Varying the number of block, 2000 hidden size, `Weather`, $T = 96$.



(b) Varying the number of blocks, 2000 hidden size, `ILI`, $T = 60$.



(c) Varying the hidden size, 3 non-linear blocks fixed, `Weather`, $T = 96$.



(d) Varying the hidden size, 1 non-linear block fixed `ILI`, $T = 60$.

Fig. 6. Model accuracy with varying number of non-linear blocks (a,b) or hidden size (c,d).

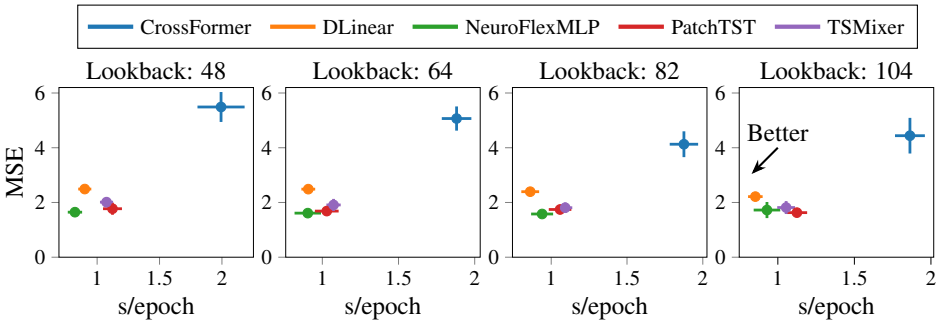
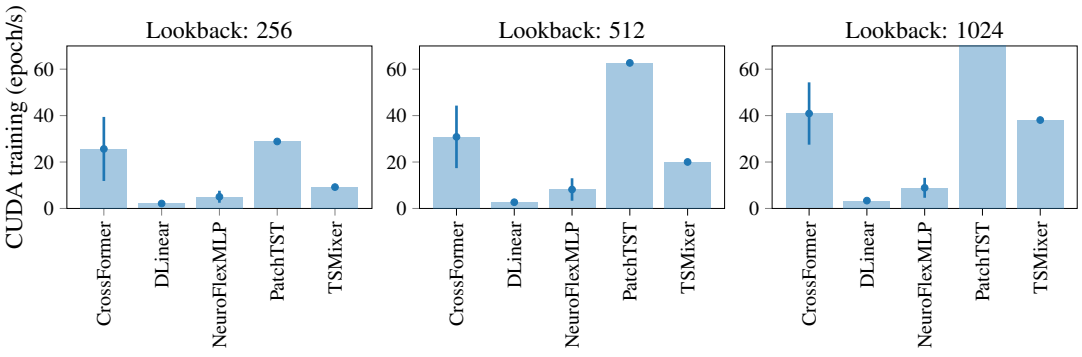
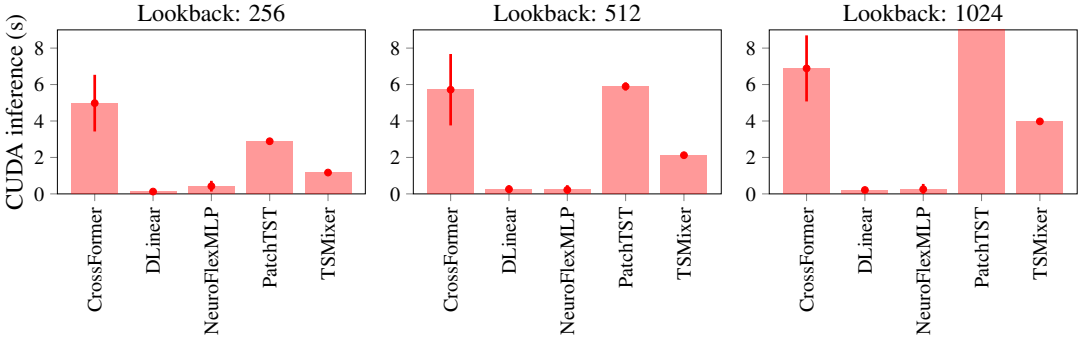


Fig. 7. Tradeoff complexity-accuracy on the `ILI` dataset. NeuroFlexMLP consistently perform as the top model considering both training time for complexity (in s/epoch) and MSE for accuracy.



(a) Training



(b) Inference

Fig. 8. Cuda training (a) and inference (b) times for the `Electricity` dataset for different lengths of input sequence. For lookback length of 1024 we encounter a CUDA out of memory error for PatchTST model that we expressed as an infinite time in the graph.

VI. Conclusions

This paper proposes an efficient MLP-based design for long-term time series forecasting tasks hinging on the capability of simple linear models to avoid overfitting. The key distinct feature of NeuroFlexMLP is the versatility across diverse time series data types leveraging non-linear operations only when necessary. Its minimal design with a few hyperparameters streamlines the hyperparameter search process. Compared to previous works, it outperforms transformer- and MLP-based baselines in several cases across datasets and settings, often being the second best performing model. NeuroFlexMLP exhibits a training cost per epoch that is nearly invariant to the input sequence length whereas transformer-based solutions scale quadratically. Its additional overhead with respect to linear-based baselines is minimal.

We demonstrated the practical relevance of these efficiency gains through a LEO satellite beam hopping use case, where NeuroFlexMLP achieved up to 35.9% MSE reduction over transformers and 28% reduction in provisioning cost. The combination of high forecasting accuracy and computationally efficient training and inference tasks makes NeuroFlexMLP a candidate for on-board deployment on resource-constrained LEO satellites, a setting where transformer-based models are impractical due to their high computational footprint.

Acknowledgments

This work has been partially funded by project Agile-6G Project (PID2024-163089NB-I00) funded by MICIU/AEI/10.13039/501100011033, 6G-ELSA (PID2022-136769NB-I00) funded by MCIN/AEI/10.13039/501100011033 FEDER, EU and project TUCAN6-CM (TEC-2024/COM-460), funded by the Madrid Regional Government (ORDEN 5696/2024). C. Fiandrino is a Ramón y Cajal awardee (RYC2022-036375-I), funded by MCIU/AEI/10.13039/501100011033 and the ESF+.

References

[1] K. Liu, J. Zhou, and D. Dong, "Improving stock price prediction using the long short-term memory model combined with online social networks," *Journal of Behavioral and Experimental Finance*, vol. 30, p. 100507, 2021.

[2] J. V. Ringwood, D. Bofelli, and F. T. Murray, "Forecasting electricity demand on short, medium and long time scales using neural networks," *Journal of Intelligent and Robotic Systems*, vol. 31, pp. 129–147, 2001.

[3] Y. Demirci, G. Mantelet, S. Martel, J.-F. Frigon, and G. K. Kurt, "Forecasting self-similar user traffic demand using transformers in LEO satellite networks," in *IEEE WiSEE*, 2025, pp. 1–6.

[4] Y. Demirci, G. Mantelet, S. Martel, J. Frigon, and G. K. Kurt, "Burst aware forecasting of user traffic demand in LEO satellite networks," *CoRR*, vol. abs/2601.14233, 2026, accepted at IEEE ICC 2026. [Online]. Available: https://arxiv.org/abs/2601.14233

[5] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, "FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting," in *Proc. of ICML*, 2022.

[6] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with Auto-Correlation for long-term series forecasting," in *Proc. of NIPS*, 2021.

[7] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. of AAAI Conference on Artificial Intelligence, AAAI*, vol. 35, 2021, pp. 11 106–11 115.

[8] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar, "Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting," in *Proc. of ICLR*, 2022.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Proc. of NIPS*, vol. 30, 2017.

[10] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?" in *Proc. of the AAAI Conference on Artificial Intelligence*, 2023.

[11] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," in *Proc. of ICLR*, 2023.

[12] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and et al, "DeepCog: Optimizing Resource Provisioning in Network Slicing With AI-Based Capacity Forecasting," *IEEE Journal on Selected Areas in Comms.*, 2020.

[13] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan, "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," *Proc. of NIPS*, vol. 32, 2019.

[14] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, and M. Long, "iTransformer: Inverted transformers are effective for time series forecasting," in *Proc. of ICLR*, 2024.

[15] S.-A. Chen, C.-L. Li, N. Yoder, S. O. Arik, and T. Pfister, "Tsmixer: An all-mlp architecture for time series forecasting," *arXiv preprint arXiv:2303.06053*, 2023.

[16] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," *arXiv preprint arXiv:1905.10437*, 2019.

[17] C. Challu, K. G. Olivares, B. N. Oreshkin, F. G. Ramirez, M. M. Canseco, and A. Dubrawski, "NHits: Neural hierarchical interpolation for time series forecasting," in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 6989–6997.

[18] P. Tang and W. Zhang, "Unlocking the power of patch: patch-based MLP for long-term time series forecasting," in *Proc. of AAAI*, 2025.

[19] A. Raman, M. Varvello, H. Chang, N. Sastry, and Y. Zaki, "Dissecting the performance of satellite network operators," *PACMNET*, vol. 1, no. CoNEXT3, pp. 15:1–15:25, 2023.

[20] Y. Li, H. Li, W. Liu, L. Liu, W. Zhao, Y. Chen, J. Wu, Q. Wu, J. Liu, Z. Lai, and H. Qiu, "A networking perspective on starlink's self-driving LEO mega-constellation," in *Proc. of ACM MobiCom*, 2023, pp. 1–16.

[21] Z. Liu, F. G. Reidys, S. Tanveer, and D. Vasisht, "Vivisecting starlink throughput: Measurement and prediction," *Proc. ACM Netw.*, vol. 3, no. CoNEXT4, pp. 1–23, 2025.

[22] Z. Lai, Z. Li, Q. Wu, H. Li, J. Li, X. Xie, Y. Li, J. Liu, and J. Wu, "Leocc: Making internet congestion control robust to LEO satellite dynamics," in *Proceedings of the ACM SIGCOMM 2025 Conference*. ACM, 2025, pp. 129–146.

[23] B. Tao, M. Masood, I. Gupta, and D. Vasisht, "Transmitting, fast and slow: Scheduling satellite traffic through space and time," in *Proc. of ACM MobiCom*, 2023, pp. 24:1–24:15.

[24] Y. Li, Y. Chen, J. Yang, J. Zhang, B. Sun, L. Liu, H. Li, J. Wu, Z. Lai, Q. Wu, and J. Liu, "Small-scale LEO satellite networking for global-scale demands," in *Proceedings of the ACM SIGCOMM 2025 Conference*. ACM, 2025, pp. 917–931.

[25] L. Izhikevich, M. Tran, K. Izhikevich, G. Akiwate, and Z. Durumeric, "Democratizing LEO satellite network measurement," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 8, no. 1, pp. 13:1–13:26, 2024.

[26] Z. Lai, H. Li, Y. Deng, Q. Wu, J. Liu, Y. Li, J. Li, L. Liu, W. Liu, and J. Wu, "Starrynet: Empowering researchers to evaluate futuristic integrated space and terrestrial networks," in *Proc. of USENIX NSDI*, 2023, pp. 1309–1324.

[27] A. Casparsen, J. E. Jakobsen, J. J. Nielsen, P. Popovski, and I. L. Mayorga, "Statistical characterization and prediction of E2E latency over LEO satellite networks," *CoRR*, vol. abs/2601.08439, 2026. [Online]. Available: https://arxiv.org/abs/2601.08439

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[29] W. A. Brock, W. D. Dechert, and J. A. Scheinkman, "A test for independence based on the correlation dimension," *Econometric Reviews*, vol. 15, no. 3, pp. 197–235, 1996.

[30] C. Fiandrino, E. P. Gómez, P. Fernández Pérez, H. Mohammadalizadeh, M. Fiore, and J. Widmer, "AIChronoLens: Advancing explainability for time series AI forecasting in mobile networks," in *Proc. of IEEE INFOCOM*, 2024, pp. 1521–1530.

[31] P. F. Pérez, I. Bravo, A. Kamath, C. Fiandrino, and J. Widmer, "ChronoProf: Profiling time series forecasters and classifiers in mobile networks with explainable AI," in *Proc. of IEEE WoWMoM*, 2025, pp. 41–50.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of IEEE CVPR*, 2016, pp. 770–778.

[33] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, and J. Choo, "Reversible instance normalization for accurate time-series forecasting against distribution shift," in *Proc. of ICLR*, 2021.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. of ACM SIGKDD*, 2019.