A Novel Hyperdimensional Computing Framework for Online Time Series Forecasting on the Edge

Mohamed Mejri [1] **Chandramouli Amarnath** [1] **Abhijit Chatterjee** [1]

Abstract

In recent years, both online and offline deep learning models have been developed for time series forecasting. However, offline deep forecasting models fail to adapt effectively to changes in time-series data, while online deep forecasting models are often expensive and have complex training procedures. In this paper, we reframe the online nonlinear time-series forecasting problem as one of linear *hyperdimensional* time-series forecasting. Nonlinear low-dimensional time-series data is mapped to high-dimensional (hyperdimensional) spaces for linear hyperdimensional prediction, allowing fast, efficient and lightweight online time-series forecasting. Our framework, *TSF-HD*, adapts to time-series distribution shifts using a novel co-training framework for its hyperdimensional mapping and its linear hyperdimensional predictor. TSF-HD is shown to outperform the state of the art, while having reduced inference latency, for both short-term and long-term time series forecasting. Our code is publicly available at: https://github.com/tsfhd2024/tsf-hd.git

1. Introduction

Time series forecasting methods have shown significant utility in fields ranging from smartgrids to traffic management (Hyndman & Athanasopoulos, 2018), and this usefulness has driven increasing research into better forecasting models (Bhatnagar et al., 2021). However, as noted in (Pham et al., 2022), naive training of deep forecasting models or offline forecasting models may not generalize well to streams of time-series data, requiring the forecaster to be trained online. Recent works such as OneNet (Zhang et al., 2023), TCN (Woo et al., 2022) and FSNet (Pham et al., 2022) have thus focused on training online deep forecasting models that aim to use novel training paradigms to enable deep neural networks to rapidly assimilate information from a time-series data stream.

However, these online deep forecasting models are expensive and difficult to deploy on edge platforms compared to linear time-series forecasters such as ARIMA (Box & Jenkins, 1968). Despite the success of deep forecasting models, it has also been shown in prior work (Zeng et al., 2023) that certain linear models such as NLinear can outperform transformer methods and deep forecasting methods in long-term time-series forecasting. Our research is thus motivated by this tradeoff between performance and overhead between linear and nonlinear models in time-series forecasting.

This work efficiently addresses that tradeoff by framing the forecasting problem as one of task-free online *hyperdimensional* learning. Hyperdimensional computing is a learning paradigm that uses high-dimensional (hyperdimensional) mappings of nonlinear input data distributions to enable linear classification or regression (Kanerva, 2009; Hernández-Cano et al., 2021) in high dimensions. This leverages the fact that functions which are nonlinear in low dimensions can be approximated as linear in high dimensions while preserving distances (Räsänen, 2015). Using this, we can run high-dimensional linear computations for nonlinear time-series forecasting, allowing rapid, low-overhead model updates from incoming samples in a data stream while maintaining state-of-the-art performance.

We thus propose *TSF-HD*, an online hyperdimensional time-series forecasting framework leveraging the rapid training and inference capabilities of linear time-series forecasting models while allowing state-of-the-art prediction accuracy. This is accomplished using a novel online training method for the hyperdimensional computing system and implementing an innovative co-training method for the high-dimensional mapping (the *encoder*) and the linear high-dimensional *regressor*. This co-training of regressor and encoder allows us to maintain the accuracy of linear predictions made by the regressor as the nonlinear time series shifts. We emphasize that TSF-HD is a task-free, online learning model - it does not require explicit detection of task shifts (changes in time series distributions, as in (Pham et al., 2022)) or time series concept drift. TSF-HD instead

[1]School of Electrical And Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia, US. Correspondence to: Mohamed Mejri <mmejri3@gatech.edu>, Chandramouli Amarnath <chandamarnath@gatech.edu>.

Preprint. Under submission.

learns online using current samples through trainable high-dimensional mappings of input data and high-dimensional linear regression. We further provide a method for autoregressive time-series hyperdimensional time-series forecasting, allowing greater accuracy over long prediction horizons or noisy time-series data by framing the problem as one of autoregressive online hyperdimensional time-series forecasting. For low-overhead, low-latency prediction we provide a sequence-to-sequence model of TSF-HD.

In summary, our work provides the following innovations: (1) *First*, we provide a novel formulation of the time series forecasting problem as one of *task free, online hyperdimensional learning*, taking advantage of the linearity of high dimensional mappings of nonlinear input data; (2) *Second*, we provide a novel online co-training framework for the encoder and regressor to enable the linearity of the mapping to be maintained as the time series evolves, without requiring explicit knowledge of task shifts. This is enhanced by the provision of an autoregressive version of TSF-HD for greater precision on noisy data streams; (3) *Lastly*, we conduct experiments against a variety of baselines to validate TSF-HD in terms of accuracy, latency and overhead.

The rest of the paper is organized as follows. Prior work is discussed in Section 2, followed by problem framing and methods in Section 3. In Section 4, we present experimental results and finally conclude in Section 5.

2. Related work

2.1. Time series forecasting (TSF)

In recent years, advancements in data availability and computational power have led to the emergence of deep learning-based techniques in time series forecasting (TSF). Traditional methods such as Autoregressive Integrated Moving Average (ARIMA) (Box & Jenkins, 1968), Autoregressive Neural Networks (AR-Net) (Triebe et al., 2019), the Holt-Winters seasonal method (Holt, 2004), and Gradient Boosting Regression Trees (GBRT) (Friedman, 2001) provide theoretical guarantees, but are typically applied to univariate time-series forecasting. Furthermore, these methods are often outperformed by deep forecasting models.

Recurrent Neural Network (RNN)-based approaches, such as those discussed by (Rangapuram et al., 2018), are examples of deep forecasting models with internal memory that are able to make predictions based on past information. However, RNN-based models face challenges due to vanishing or exploding gradient problems and inefficient training procedures. Transformers, exemplified by the Informer model (Zhou et al., 2021), have outperformed the RNN paradigm, leveraging the self-attention mechanism's capability to capture correlations in temporal sequences. SCINet (Liu et al., 2022) is a convolutional deep forecaster that recursively downsamples and convolves data to extract complex temporal features for accurate time-series forecasting. Despite the success of transformers in time series forecasting, simpler linear models like NLinear (Zeng et al., 2023) have outperformed transformer methods in some long-term time series forecasting applications in terms of accuracy.

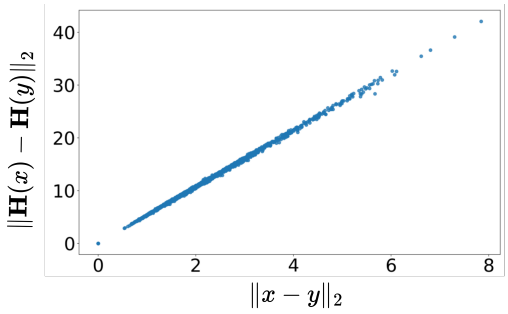
However, none of the methods described above are adapted to online time series forecasting. Recent work in online time series forecasting includes an online Temporal Convolutional Network (TCN) with 10 hidden layers that successfully captures periodic patterns (Woo et al., 2022). The Experience Replay method (ER) (Chaudhry et al., 2019) enhances online TCN by adding episodic memory to mix old and new learning samples, improving generalization. DER++ (Buzzega et al., 2020) augments the standard ER with an ℓ_2 knowledge distillation loss on the previous logits to align the network's logits, ensuring consistency with its past behavior. FSNet (Pham et al., 2022), or Fast and Slow Learning Network, combines rapid adaptation to new data with memory recall of past events to enable adaptation to task shifts (concept drift) in the time series. OneNet (Zhang et al., 2023) extends the FSNet implementation by integrating and updating two models in real-time: one concentrates on modeling dependencies over time, and the other focuses on dependencies across variables. We note that these online deep forecasters are expensive and cumbersome to train, and may not be applicable to low-latency forecasting in a shifting nonlinear time series stream.

Using high-dimensional (hyperdimensional) mappings (encodings) to forecast a nonlinear data stream using linear hyperdimensional regression has not been explored in prior work, primarily due to the lack of a co-trainable encoder and regressor in state of the art hyperdimensional regressors like RegHD (Hernández-Cano et al., 2021; Chen et al., 2022). This deficiency prevents them from adapting to time-series task shifts and learning online in an effective manner.

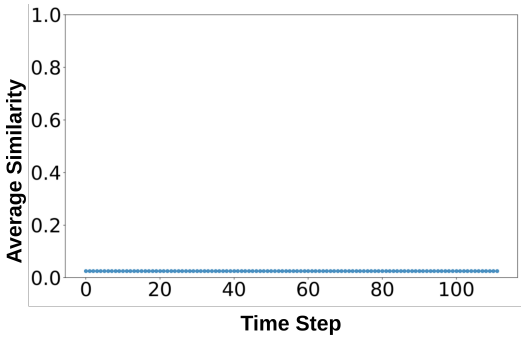
2.2. Online learning

Online learning (Lopez-Paz & Ranzato, 2017) aims to learn several tasks sequentially with limited access to past experiences. A good learner is one that achieves the best trade-off between adaptation to new tasks and maintenance of past knowledge from previous tasks, a tradeoff known as the stability-plasticity dilemma (Grossberg & Grossberg, 1982). One popular framework is the complementary learning system (CLS) (McClelland et al., 1995; Kumaran et al., 2016). Continual deep learning methods using the CLS framework augment slow deep learning algorithms with the ability to learn quickly from a data stream, either through experience replay (Lin, 1992; Riemer et al., 2018; Rolnick et al., 2019; Aljundi et al., 2019; Buzzega et al., 2020) or via fast and slow learning components (Pham et al., 2020; Arani et al.,

2022; Pham et al., 2022).



(a) Distance preservation across hyperdimensional mappings of input sequences $\mathbf{H}(x)$.



(b) The mean cosine similarity between rows of the encoder matrix W_e used to map low-dimensional input sequences to the hyperdimensional space is seen to be very low as the time series evolves (t increases), preserving orthogonality while updating W_e online.

Figure 1. Distance preservation (linear relationship between distances) and orthogonality of the different components of the trainable encoder matrix W_e across hyperdimensional mappings of input sequences $\mathbf{H}(x) = x.W_e$ (ETTm1 dataset), showing preservation of the properties detailed in (Räsänen, 2015).

3. Online Time Series Forecasting

3.1. Problem Framing and Preliminaries

In this paper, we focus on the *online* multivariate time series forecasting (TSF) problem. Given a long sequence $\mathbf{X}^* \in \mathbb{R}^{N \times L}$ where L is the length of the sequence $\mathbf{X}^*$ and N is the number of variables in the time series $\mathbf{X}^*$ (the dimension of the input $x \in \mathbf{X}^*$) and given a look back window of fixed length T, at timestamp t, the time series forecasting task is to predict $X_{t+1:t+\tau} = [x_{t+1}, ..., x_{t+\tau}]$ based on the past T steps $X_{t-T+1:t} = [x_{t-T+1}, ..., x_t] \in \mathbb{R}^{N \times T}$. Here τ is the referred to as the *forecast horizon*. Each data sample $x \in \mathbb{R}^N$. Online learning assumes that data arrives in a stream, drawn from a shifting distribution. Our approach leverages the ability of hyperdimensional spaces to represent nonlinear input data in low dimensions as linear in high dimensions (Räsänen, 2015). The learner is not aware when the data changes in trend (i.e, moves from one task to another) making the online time series forecasting problem one of task-free online learning. We thus use hyperdimensional regression with a trainable hyperdimensional mapping (the *encoder*) and a novel co-training framework for the encoder and the linear hyperdimensional regressor.

Hyperdimensional (HD) computing represents data using extremely high dimensional spaces (Kanerva, 2009) referred to as 'hyperspaces'. A vector in this space is referred to as a 'hypervector'. Multivariate hyperdimensional regression in TSF-HD involves three primary phases:

Hypervector Encoding: In this phase, an input data sequence $X \in \mathbb{R}^{N \times T}$ is transformed from the feature space $\mathcal{X}$ to a hyperspace $\mathcal{H}$ ($dim(\mathcal{H}) = D \gg T$) using a function $\mathbf{H} : \mathcal{X} \rightarrow \mathcal{H}$. The encoding can be accomplished using various methods such as N-gram based encoders (Imani et al., 2018), or linear projection (Dutta et al., 2022). Our work uses a *trainable* encoder consisting of a linear mapping followed by a ReLU function, yielding

$$\mathbf{H}(X) = \mathbb{1}_{X>0}[X.W_e + b_e] \tag{1}$$

where $W_e \in \mathbb{R}^{T \times D}$ is a matrix whose rows consist of hypervectors mapping the input X to different components of the HD space, $\mathbb{1}_{X>0}(.)$ denotes the ReLU function and $b_e \in \mathbb{R}^D$ is a trainable bias added to each row of $X.W_e$.

As per (Räsänen, 2015), based on the Johnson-Lindenstrauss lemma, the distances between x and y ($x, y \in \mathcal{X}$) are preserved under the encoder mapping $H(.)$ within a scaling factor for a *non-trainable*, randomized projection matrix mapping $\mathcal{X}$ to $\mathcal{H}$, allowing an HD system to take advantage of the linearity of high dimensional mappings of lower dimensional nonlinear input data. To enable forecasting on a shifting time series, our *trainable* encoder system evolves with the time series sequence while still preserving distances, as seen in Figure 1a, where the plot between the distances $\|x - y\|_2$ and $\|\mathbf{H}(x) - \mathbf{H}(x)\|_2$ is linear, where $\|.\|_2$ denotes the L2 norm. Similarly, as per (Räsänen, 2015), for a *non-trainable* randomized projection matrix mapping $\mathcal{X}$ to $\mathcal{H}$, the hypervectors that make up the matrix are ideally orthogonal to one another. We see that this approximately holds through the time series in Figure 1b for our trainable matrix W_e, with the average cosine similarity between its rows remaining near-zero. Further validation of distance preservation and orthogonality of W_e can be found in Appendix A. Hyperdimensional computing using our novel trainable encoder and regressor formulation thus allows us to take advantage of the desirable properties of hyperdimensional encoding detailed in (Räsänen, 2015) while continually updating our model as the data stream shifts.

HD System Training: For our linear HD regressor, the goal is to find an approximation $\tilde{X}_{t+1:t+\tau}$, given an *encoded* input sequence taken from the lookback window, $\mathbf{H}(X_{t-T+1:t})$ to minimize a loss function $\mathcal{L}(.)$ calculated

from the regression error across the prediction horizon ($\tilde{X}_{t+1:t+\tau} - X_{t+1:t+\tau}$). This involves a trainable regressor hypervector or matrix, denoted as W_r, and a trainable regressor bias b_r. W_r and b_r are updated in conjunction with W_e and b_e to minimize $\mathcal{L}(\tilde{X}_{t+1:t+\tau} - X_{t+1:t+\tau})$, using online gradient descent (OGD) with the AdamW optimizer.

Our input data is not normalized or standardized, simulating a real-time online learning environment, and data point values may vary significantly. An L2-norm-based loss function could lead to rapid divergence of the loss due to such variations. While an L1 norm is more suitable, its non-differentiability around 0 presents a challenge. Therefore, we opt for the *Huber loss* ($\mathcal{L}_H$), as detailed for a single prediction step ($\tau = 1$) in Eq. 2. We denote $\Delta x_{t+1} = x_{t+1} - \tilde{x}_{t+1}$, yielding:

$$\mathcal{L}_H(x_{t+1}, \tilde{x}_{t+1}) = \begin{cases} \frac{1}{2}\|\Delta x_{t+1}\|_2, & \text{if } |\Delta x_{t+1}| \leq 1 \\ \|\Delta x_{t+1}\|_1 - \frac{1}{2}, & \text{otherwise} \end{cases} \tag{2}$$

The total loss is thus

$$\mathcal{L}(x_{t+1}, \tilde{x}_{t+1}) = \mathcal{L}_H(x_{t+1}, \tilde{x}_{t+1}) + \mathcal{R}(W_e, W_r, b_e, b_r) \tag{3}$$

where $\mathcal{R}(.)$ is an L2 norm regularization function.

HD Inference The prediction $\tilde{X}_{t+1:t+\tau}$ is generated after mapping the input samples $X_{t-T+1:t}$ to the hyperspace $\mathcal{H}$ as in Equation 1. A single-step prediction $\tilde{x}_{t+1}$ is then computed as:

$$\tilde{x}_{t+1} = \mathbf{R}(\mathbf{H}(X_{t-T+1:t})) = [((\mathbf{H}(X_{t-T+1:t}))_1 .W_r, ... \\ ..., (\mathbf{H}(X_{t-T+1:t}))_N .W_r] + b_r \tag{4}$$

where $\mathbf{H}(.)$ is the encoding function of Equation 1 and $\mathbf{R}(.)$ is the regression function that uses the regression matrix W_r and bias b_r. $(\mathbf{H}(X_{t-T+1:t}))_i$ denotes the ith row of an $N \times D$-dimensional encoded input sequence. The regressor function for the ith element of the predicted vector ($1 \leq i \leq N$) is thus $(\tilde{x}_{t+1})_i = \mathbf{R}_i(\mathbf{H}(X_{t-T+1:t})) = (\mathbf{H}(X_{t-T+1:t}))_i .W_r + b_r$

We provide two distinct TSF-HD frameworks: the autoregressive AR-HDC and the sequence-to-sequence Seq2Seq-HDC. AR-HDC is more accurate than Seq2Seq-HDC, but incurs higher overhead and is slower for long-term forecasting than Seq2Seq-HDC.

3.2. TSF-HDC Frameworks

3.2.1. Overview

Figure 2 presents an overview of both the autoregressive version of our framework, called AR-HDC, and the sequence-to-sequence version, called Seq2Seq-HDC. Both models use T multivariate past samples $X_{t-T+1...t} \in \mathbb{R}^{N \times T}$, to predict the next τ samples, $\tilde{x}_{t+1}$ to $\tilde{x}_{t+\tau}$.

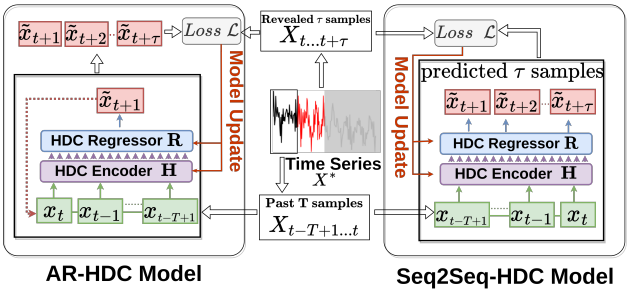


Figure 2. Autoregressive (AR-HDC) & Sequence-to-Sequence (Seq2Seq) frameworks for TSF-HD. Our novel co-training system jointly updates the encoder $\mathbf{H}(.)$ and regressor $\mathbf{R}(.)$ online.

The autoregressive AR-HDC framework (Figure 2, left) predicts one time step ahead at a time. This prediction is then used to make further predictions, as seen in Figure 2 when $\tilde{x}_{t+1}$ is fed back - the system thus predicts $\tilde{x}_{t+1}$ using $X_{t-T+1:t}$, then predicts $\tilde{x}_{t+2}$ using $[X_{t-T+2:t}, \tilde{x}_{t+1}]$, until it predicts $\tilde{x}_{t+\tau}$ using $[X_{t-T+\tau+1:t}, \tilde{X}_{t+1:t+\tau-1}]$. Once the prediction is complete, the system uses the true values of the τ samples (x_{t+1} to $x_{t+\tau}$) to update the model - model updates are thus done periodically once new information is known. AR-HDC then attempts to minimize the loss $\mathcal{L}$ by updating the weights of its components (the regressor and encoder), as in Figure 2. Unlike prior work, the regressor and encoder are updated *jointly* online to minimize $\mathcal{L}$.

In contrast, the Seq2Seq-HDC system (Figure 2, right) predicts the future sample values (from $\tilde{x}_{t+1}$ to $\tilde{x}_{t+\tau}$) in one shot using the values of $X_{t-T+1:t}$, as seen in Fig. 2. This is faster than the iterative prediction of AR-HDC, but may not be as accurate. As seen in Figure 2, the true values of the time points in the prediction horizon (ranging from $t+1$ to $t+\tau$) are then used to calculate the prediction loss across the sequence and update the model, again updating regressor and encoder jointly (co-training) to minimize $\mathcal{L}$.

3.2.2. Autoregressive HDC (AR-HDC)

AR-HDC predicts one element ahead at a time, modeling the time series as an autoregressive process. The classical autoregressive predictor yields the predicted multivariate sample $\tilde{x}_{t+1} \in \mathbb{R}^N$ using a weighted average of past T (i.e, the look back window value) samples $X_{t-T+1:t}$, yielding $\tilde{x}_{t+1} = \sum_{j=0}^{T} w_j .x_{t-j+T}$.

For AR-HDC, the weighted average of x_{t-j+T} of the previous equation is replaced with an hyperdimensional encoder and regressor (rewriting Equation 4):

$$(\tilde{x}_{t+1})_i = < W_r, (\mathbf{H}(X_{t-T+1:t}))_i > + b_r \tag{5}$$

where $< . >$ denotes the inner product operation, $W_r \in \mathbb{R}^D$ denotes the trainable regressor hypervector (described in Section 3.1), $(\tilde{x}_{t+1})_i$ is the ith term of the predicted vector ($1 \leq i \leq N$), b_r is the regressor bias term and

$\mathbf{H}(X_{t-T+1:t})_i$ denotes ith row of the encoded input sequence $X_{t-T+1:t}$. AR-HDC then adds $\tilde{x}_{t+1}$ to its lookback window and removes the last term from the lookback window to predict x_{t+2} similarly, using the sample vector $[X_{t-T+2:t}, \tilde{x}_{t+1}]$. This is described in Algorithm 1.

Algorithm 1 Autoregressive HDC (AR-HDC)

1: $X \leftarrow X_{t-T+1:t} = [x_{t-T+1}, \ldots, x_{t-1}, x_t]$
2: **for** $i = 1$ **to** τ **do**
3: $\quad \tilde{x}_{t+i} \leftarrow \mathbf{R}(\mathbf{H}(X))$
4: $\quad X \leftarrow (X \setminus \{x_{t-T+i}\}) \cup \{\tilde{x}_{t+i}\}$
5: **end for**
6: **for** $i = 1$ **to** τ **do**
7: $\quad$ OGD$(\mathbf{H}, \mathbf{R}, \mathcal{L}(\tilde{x}_{t+i}, x_{t+i}))$ ▷ Training Step
8: **end for**

The procedure for AR-HDC consists of two phases: a prediction phase for τ future samples (from line 2 to line 5 of Algorithm 1) and an online learning phase (from line 6 to line 8) to update the encoder and regressor once the true values of $X_{t+1:t+\tau}$ are known. At line 3, AR-HDC employs the encoding $\mathbf{H}$, followed by the regression function $\mathbf{R}$, to predict the i^{th} future sample based on the past sequence X, which is initialized in line 1 as the set of samples $X = X_{t-T+1:t}$. In line 4, the first element x_{t-T+i} (initially x_{t-T+1}) of the sequence X is removed, and the predicted sample $\tilde{x}_{t+i}$ is appended. This updated sequence is then fed forward to the encoder and regressor in line 2. This is repeated until $x_{t+\tau}$ is predicted from $X = [X_{t-T+\tau+1:t}, \tilde{X}_{t+1:t+\tau-1}]$. Following the prediction phase, the system parameters (encoder weights and regression hypervectors) are updated using online gradient descent as in Section 3.1. The overhead of AR-HDC thus scales linearly with τ. For large τ, a sequence-to-sequence predictor may thus be more practical.

3.2.3. Sequence-To-Sequence HDC (Seq2Seq-HDC)

The Seq2Seq-HDC algorithm is an efficient and straightforward algorithm for time series forecasting, where the past T steps of the sequence are encoded into a $D \gg T$-dimensional hyperspace and a linear HDC regressor consisting of a $D \times \tau$-dimensional regression matrix and trainable bias b_r is used. The HDC encoder $\mathbf{H}$ is identical to that of AR-HDC. The system begins by generating hypervector encodings $h = \mathbf{H}(X_{t-T+1:t})$, which are then used by the regressor $\mathbf{R}(.)$:

$$\tilde{X}_{t+1:t+\tau} = \mathbf{R}(h) = h.W_r + b_r \tag{6}$$

Here, $X_{t-T+1..t} \in \mathbb{R}^{N \times T}$ is the input sequence of past elements. It is encoded into an hyperdimensional structure $h \in \mathbb{R}^{N \times D}$ using W_e and b_e as in Equation 1. $W_r \in \mathbb{R}^{D \times \tau}$ is the HDC regression matrix for the output sequence $\tilde{X}_{t+1:t+\tau} \in \mathbb{R}^{N \times \tau}$ and $b_r \in \mathbb{R}^{\tau}$ is the regression bias. This process is further described in Algorithm 2.

Algorithm 2 Seq2Seq-HDC Algorithm

1: **Initialization:**
2: $X_{t-T+1..t} \leftarrow [x_{t-T+1}, \ldots, x_{t-1}, x_t]$
3: **Forward Pass:**
4: $h \leftarrow \mathbf{H}(X)$
5: $\tilde{X}_{t..t+\tau} \leftarrow \mathbf{R}(h)$
6: **Online Learning Phase:**
7: OGD$((\mathbf{H}, \mathbf{R}, \mathcal{L}(X_{t..t+\tau}, \tilde{X}_{t..t+\tau})))$ ▷ Training Step

In line 4 the input, consisting of T past elements $X_{t-T+1:t}$ are encoded into high dimensional vectors $h \in \mathbb{R}^{N \times D}$ using the encoder $\mathbf{H}$. In line 5, the hypervectors h are forwarded to the regressor to generate the predicted τ future points $\tilde{X}_{t+1:t+\tau}$ using $\mathbf{R}(.)$. In the line 7, the actual values $X_{t+1:t+\tau}$ are used to update the system weights (encoder and all regressor matrices and biases) in order to minimize the loss $\mathcal{L}(X_{t+1:t+\tau}, \tilde{X}_{t+1:t+\tau})$ via online gradient descent.

4. Experiments

4.1. Experimental settings

4.1.1. Datasets & metrics

We empirically validate TSF-HD on eight real-world benchmark datasets (ETTh1 and ETTh2 (hourly electric transformer data), ETTm1 and ETTm2 (minute-by-minute electric transformer data), WTH (Weather forecasting), ECL (hourly electricity consumption), Exchange (currency exchange rates) and ILI (Influenza-like illness occurrence)), the details of which are in Appendix B.1. For short-term Time Series Forecasting (TSF), all eight datasets were utilized for evaluation. For long-term TSF, all datasets except ETTh2 and ETTm2 were used. We also use a synthetic abrupt dataset (called *S-A*), derived from (Pham et al., 2022), to examine speed of adaptation to time series task shifts. This univariate dataset ($N = 1$) contains abrupt and recurrent components, where samples switch between different autoregressive (AR) processes.

To evaluate model precision we use the Root Relative Squared Error (RSE) and Empirical Correlation Coefficient (CORR) metrics, following (Lai et al., 2018). Details of these metrics are available in Appendix B.2. To evaluate model overhead and efficiency we record inference latency in seconds and power use (on edge platforms) in watts (W).

4.1.2. Model baselines & Implementation Setup

We compare TSF-HD to several online learning baselines (*FSNet* (Pham et al., 2022), *ER*(Chaudhry et al., 2019), *DER++* (Buzzega et al., 2020), *OnlineTCN* (Woo et al.,

		Seq2Seq-HDC		AR-HDC		Fsnet		ER		DR++		OnlineTCN		Informer	
	τ	*RSE*	*CORR*	*RSE*	*CORR*	*RSE*	CORR	*RSE*	*CORR*	*RSE*	*CORR*	*RSE*	*CORR*	*RSE*	*CORR*
Exchange	3	**0.032**	**0.999**	0.033	**0.999**	0.339	0.916	0.14	0.99	0.109	0.995	0.194	0.973	0.682	0.813
	6	0.043	**0.999**	**0.034**	**0.999**	0.516	0.835	0.148	0.987	0.159	0.987	0.175	0.98	0.704	0.758
	12	0.053	0.998	**0.035**	**0.999**	0.497	0.872	0.171	0.982	0.188	0.979	0.274	0.95	0.758	0.754
ECL	3	0.314	0.952	0.288	0.966	0.231	0.966	0.194	0.978	**0.181**	**0.981**	0.218	0.97	0.998	0.153
	6	**0.153**	**0.988**	0.305	0.976	0.223	0.971	0.174	0.977	0.168	0.982	0.186	0.977	0.998	0.137
	12	**0.106**	**0.995**	0.121	0.994	0.218	0.979	0.137	0.992	0.136	0.992	0.146	0.991	0.999	0.046
ETTh2	3	0.142	0.989	**0.128**	**0.994**	0.155	0.993	0.141	0.993	0.136	**0.994**	0.164	0.992	0.751	0.838
	6	0.162	0.987	0.177	**0.991**	0.191	0.989	0.158	**0.991**	**0.154**	**0.991**	0.191	0.986	0.757	0.835
	12	0.178	0.984	**0.172**	**0.988**	0.229	0.983	0.197	0.986	0.188	0.987	0.226	0.983	0.753	0.834
ETTh1	3	0.389	**0.928**	**0.349**	0.92	0.509	0.907	0.652	0.821	0.513	0.877	0.583	0.871	0.947	0.658
	6	0.482	0.894	**0.443**	0.875	0.609	0.882	0.489	0.906	0.459	**0.917**	0.538	0.899	0.99	0.652
	12	**0.371**	**0.938**	0.448	0.902	0.727	0.843	0.533	0.896	0.521	0.899	0.597	0.878	1.005	0.646
ETTm1	3	0.111	**0.994**	**0.11**	**0.994**	0.198	0.987	0.351	0.914	*0.563*	*0.821*	0.621	0.892	0.501	0.887
	6	**0.135**	**0.991**	0.148	0.99	0.223	0.985	0.335	0.954	0.299	0.952	0.353	0.911	0.506	0.884
	12	**0.181**	**0.984**	0.209	0.982	0.249	0.981	0.276	0.965	0.286	0.956	0.332	0.941	0.509	0.883
ETTm2	3	0.104	0.994	**0.086**	**0.998**	0.143	0.995	0.149	0.983	*0.135*	*0.986*	0.286	0.951	0.911	0.795
	6	0.136	0.991	**0.114**	**0.996**	0.163	0.994	0.135	0.992	0.141	0.991	0.177	0.991	0.766	0.904
	12	**0.161**	0.987	0.171	**0.993**	0.155	0.992	0.162	0.992	0.148	0.992	0.174	0.991	0.781	0.901
WTH	3	0.671	0.784	0.642	0.797	0.719	0.811	0.694	0.817	0.682	0.819	0.719	0.809	**0.58**	**0.831**
	6	0.712	0.773	0.674	0.785	0.757	0.807	0.708	0.819	0.697	**0.822**	0.732	0.812	**0.612**	0.811
	12	0.665	0.799	**0.651**	0.802	0.784	0.8	0.732	0.815	0.722	**0.818**	0.759	0.808	0.655	0.772
ILI	3	0.203	0.979	**0.135**	**0.998**	0.143	0.996	0.151	0.996	0.148	0.997	0.158	0.996	1.187	0.348
	6	0.247	0.969	0.202	**0.995**	0.202	0.993	0.199	0.994	**0.189**	0.994	0.199	0.994	1.168	0.373
	12	0.296	0.956	0.297	**0.989**	**0.248**	**0.989**	0.268	0.987	0.328	0.985	0.279	0.987	1.066	0.337

Table 1. Short Term Time Series Forecasting Performance of AR-HDC & Seq2Seq-HDC compared to the baseline. We report the mean of RSE and CORR of the experiments. The results in bold are the best and in blue and underlined are second best

2022)), transformers (*Informer* (Zhou et al., 2021)), convolutional TSF models (*SCINet* (Liu et al., 2022)), linear TSF models (*NLinear* (Zeng et al., 2023)), a naive direct multi-step that repeats the value of the look-back window (*Repeat*) and Gradient Boosting Regression Trees (*GBRT* (Friedman, 2001)). Further details of the baselines can be found in Appendix C. For brevity, in the main body of the paper we present the online learning and transformer baselines. TSF-HD sees similar effectiveness when compared to the other baselines, as shown in Appendix D.

Similar to prior work (Pham et al., 2022), each time-series forecasting scenario is split into: (1) A *warmup* phase where the model learns from a small portion of the dataset without measuring model performance, and (2) An *online learning* phase where model predictions are evaluated and the model parameters are updated using the ground truth data as it is made available. In all scenarios, the trainable encoder weight matrix W_e and bias b_e and the trainable regressor hypervectors W_r and bias b_r are initially sampled from the uniform distribution with the interval $\left[\frac{-1}{T}, \frac{1}{T}\right]$ before training begins. TSF-HD is trained periodically, predicting the next τ samples and then updating model weights when the system reveals those points, ensuring no overlap between predictions. NLinear, GBRT, Informer and SCINet are not meant to be trained online. Following (Pham et al., 2022), we thus trained them only in the warm-up phase for 10 epochs. For these baselines, the training/validation set proportion is 75:25. For the online learning baselines the warm-up/online learning phase proportion is 25:75.

We fix the hypervector dimension at $D = 1000$. The look-back window is fixed to twice the forecast horizon: $T = 2\tau$. For short term TSF, $\tau \in \{3, 6, 12\}$ and for the long term TSF, $\tau \in \{96, 192, 384\}$ for all datasets except ILI where $\tau \in \{24, 36, 48\}$ and Exchange where $\tau \in \{96, 192, 224\}$. The experiments are run five times with different random seed values, with the mean RSE and CORR reported.

4.2. Performance Analysis

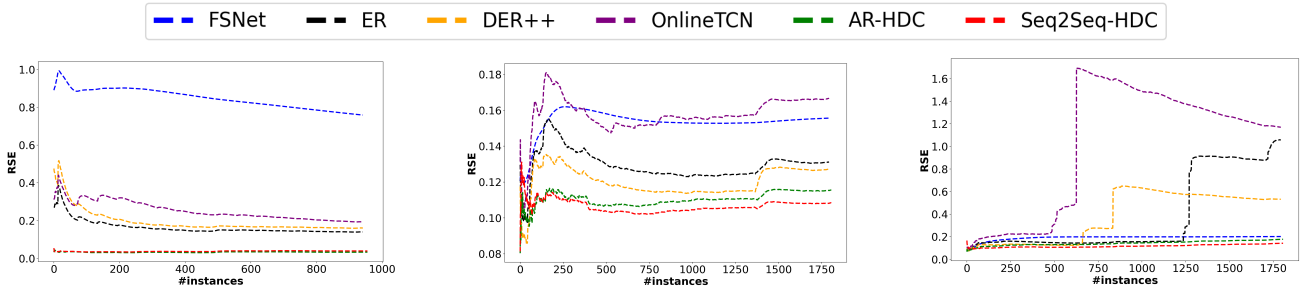
4.2.1. Model Precision

Table 1 shows the RSE and CORR for the two TSF-HD frameworks (AR-HDC and Seq2Seq-HDC) compared to the baselines. For short-term TSF, AR-HDC surpasses all state-of-the-art (SOTA) models all 24 test cases with either lower RSE and higher CORR. Seq2Seq-HDC shows comparable performance, outshining SOTA models in 10 out of 24 test cases, and is the top algorithm in 2 of these cases. We thus see that for a *majority* of the test cases in short-term TSF, TSF-HD frameworks outperform the state of the art.

Table 2 presents validation results for long-term TSF, comparing the TSF-HD frameworks to the state of the art. We see that both TSF-HD frameworks perform better in comparison to the state of the art for the long-term as opposed to the short-term TSF cases. AR-HD outperforms the baseline models in either CORR or RSE in 16 out of 18 test cases. Seq2Seq-HDC achieves the best results in 4 out of 18 cases for RSE and ranks second in 10 of the 18 cases. Notably, AR-HDC demonstrates impressive performance in the ETTh1, ETTm1, ILI, and WTH datasets for very long forecast horizons ($\tau = 384$). Due to the high value

		Seq2Seq-HDC		AR-HDC		Fsnet		ER		DR++		OnlineTCN		Informer	
	τ	*RSE*	*CORR*	*RSE*	*CORR*	*RSE*	CORR	*RSE*	*CORR*	*RSE*	*CORR*	*RSE*	*CORR*	*RSE*	*CORR*
ETTh1	96	0.601	**0.846**	**0.568**	0.84	1.073	0.722	0.864	0.778	0.871	0.778	0.897	0.763	1.397	0.629
	192	0.678	0.803	**0.599**	**0.827**	1.353	0.626	1.089	0.696	1.066	0.703	1.118	0.675	1.455	0.627
	384	0.797	0.739	**0.628**	**0.831**	1.228	0.616	1.347	0.609	1.295	0.617	1.347	0.604	1.508	0.625
ETTm1	96	0.315	0.951	**0.297**	**0.965**	0.355	0.951	0.347	0.953	0.341	0.956	0.359	0.951	0.822	0.856
	192	0.392	0.925	0.373	**0.949**	0.409	0.937	0.374	0.942	**0.372**	0.943	0.376	0.941	0.855	0.862
	384	0.465	0.893	**0.431**	0.924	0.522	0.915	0.454	0.926	0.448	**0.927**	0.476	0.925	0.841	0.859
ILI	24	0.288	0.957	**0.229**	**0.986**	0.304	0.967	0.382	0.936	0.357	0.947	0.411	0.932	1.09	0.211
	36	0.391	0.92	**0.199**	**0.99**	0.389	0.945	0.454	0.931	*0.489*	*0.926*	0.477	0.927	1.122	0.221
	48	0.396	0.919	**0.204**	**0.989**	0.503	0.916	0.534	0.893	0.553	0.895	0.582	0.879	0.193	0.144
Exchange	96	0.211	0.978	**0.097**	**0.997**	2.83	0.373	1.353	0.586	1.154	0.641	2.241	0.395	0.904	0.577
	192	0.318	0.948	**0.155**	**0.994**	2.832	0.254	2.311	0.252	2.19	*0.254*	2.993	0.203	0.968	0.398
	224	0.328	0.945	**0.137**	**0.995**	3.697	0.162	2.204	0.261	2.106	0.265	2.436	0.215	0.989	0.335
ECL	96	**0.177**	0.985	0.206	**0.989**	0.388	0.947	0.221	0.982	0.215	0.982	0.219	0.978	1	0.027
	192	**0.214**	0.977	0.294	**0.979**	0.413	0.913	0.303	0.952	0.306	0.951	0.305	0.951	1	0.031
	384	**0.282**	0.962	0.302	**0.979**	0.517	0.861	0.504	0.866	0.484	0.866	0.521	0.857	1	0.077
WTH	96	0.697	0.793	**0.633**	**0.796**	0.886	0.766	0.807	0.787	0.798	0.789	0.838	0.782	0.841	0.567
	192	0.723	0.784	**0.632**	**0.791**	0.911	0.753	0.854	0.771	0.845	0.772	0.926	0.759	0.895	0.549
	384	0.777	0.762	**0.653**	**0.776**	0.965	0.726	0.936	0.743	0.894	0.749	1.014	0.732	0.871	0.547

Table 2. Long Term Time Series Forecasting Performance of AR-HDC & Seq2Seq-HDC compared to the baseline. We report the mean of RSE and CORR of the experiments. The results in bold are the best and in blue and underlined are second best



(a) Exchange (b) ETTm2 (c) ETTm1

Figure 3. Evolution of the cumulative RSE during online learning with forecasting window $\tau = 6$

of D compared to τ, TSF-HD frameworks are thus able to perform accurate hyperdimensional linear regression after encoding the input data, leading to this high performance.

The high values of RSE seen for the baselines in Tables 1 and 2 (and for offline predictor data in Appendix D), particularly for long-term TSF, indicate a collapse in performance compared to a naive predictor. This is in part due to increased values of τ and in part due to the fact that our experiments use raw data without standardization or normalization, since in an online learning scenario with shifting data streams we do not know the mean and standard deviation of input data.

4.2.2. Convergence of TSF-HD Models

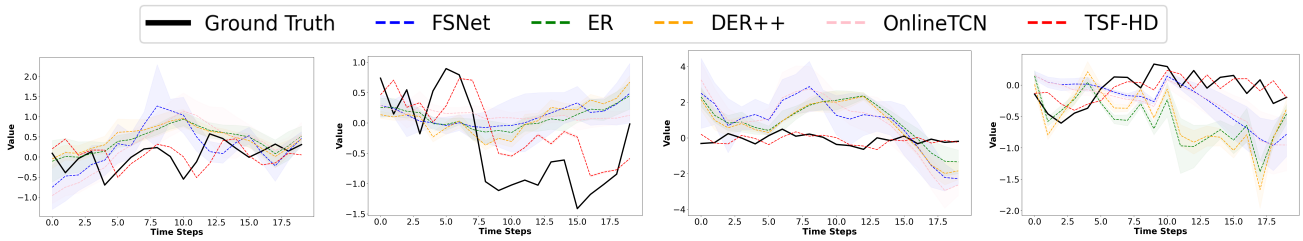
Figure 3 shows the cumulative average RSE of the online learning baselines for short-term TSF ($\tau = 6$) compared to AR-HDC and Seq2Seq-HDC on the Exchange, ETTm1 and ETTm2 datasets.

We see that in the first 20% of the data (Exchange & ETTm2), all models suffer from concept drift and attempt to adapt to the shift in trend. The remainder of the data appears stationary, as indicated by the relative convergence of the RSE for all test cases. For all test cases, AR-HDC outperforms all other models, achieving faster RSE convergence (quicker adaptation to task shifts) and maintaining a generally lower RSE value (efficient adaptation).

The Seq2Seq-HDC model's RSE convergence follows a similar trend to AR-HDC for ETTm1 and Exchange. For all the test cases, the performance (i.e, speed of convergence and asymptotic value) of AR-HDC and Seq2Seq-HDC are comparable except for ETTm2, where Seq2Seq-HDC a has lower final RSE value. In the ETTm1 case, OnlineTCN, ER and DER++ are highly sensitive to shifts. These observations validate our framing of the TSF problem as one of online linear hyperdimensional forecasting, using a trainable mapping from low (input) dimensions to the hyperspace.

4.2.3. Adaptation to Task Shifts

To analyze the TSF-HD's adaptation to task shifts in data streams, we use the abrupt synthetic dataset S-A (Pham et al., 2022), composed of different tasks concatenated together. Thus, we begin by running warm-up training on a warm-up autoregressive (AR) process in S-A for 1000 time steps.



(a) 1^{st} task shift zone (b) 2^{nd} task shift zone (c) 3^{rd} task shift zone (d) 4^{th} task shift zone

Figure 4. Visualization of the baseline and TSF-HD method performance as the tasks shift between four different autoregressive processes.

		Latency				*Power*			
	τ	AR-HDC	Seq2Seq-HDC	FSNet	OnlineTCN	AR-HDC	Seq2Seq-HDC	FSNet	OnlineTCN
Exchange	3	**0.016**$^{\pm 0.002}$	0.066$^{\pm 0.013}$	0.452$^{\pm 0.121}$	0.213$^{\pm 0.079}$	**3.395**$^{\pm 0.534}$	4.246$^{\pm 0.945}$	4.855$^{\pm 0.635}$	4.654$^{\pm 0.748}$
	96	0.937$^{\pm 0.347}$	**0.073**$^{\pm 0.041}$	0.538$^{\pm 0.227}$	0.273$^{\pm 0.093}$	5.257$^{\pm 0.511}$	**4.437**$^{\pm 0.855}$	4.975$^{\pm 0.642}$	4.788$^{\pm 0.804}$
ETTh1	3	**0.0144**$^{\pm 0.001}$	0.065$^{\pm 0.011}$	0.47$^{\pm 0.122}$	0.212$^{\pm 0.078}$	**3.508**$^{\pm 0.486}$	4.236$^{\pm 0.904}$	4.766$^{\pm 0.637}$	4.567$^{\pm 0.783}$
	96	0.867$^{\pm 0.03}$	**0.071**$^{\pm 0.038}$	0.530$^{\pm 0.209}$	0.295$^{\pm 0.125}$	5.286$^{\pm 0.203}$	**4.419**$^{\pm 0.967}$	4.961$^{\pm 0.591}$	4.858$^{\pm 0.698}$

Table 3. Power consumption & latency of two online learning baselines and the TSF-HD model on RaspberryPI ($mean^{\pm std}$)

Dimension			0.5k	1k	5k	10k
RSE	**Seq2Seq-HDC**	ETTh2	0.971	0.756	0.164	0.143
		ETTm2	0.849	0.349	0.179	0.178
	AR-HDC	ETTh2	0.059	0.059	0.059	0.058
		ETTm2	0.047	0.049	0.052	0.053

Table 4. Effect of hypervector dimensionality reduction on the RSE of ETTh2 & ETTm2 datasets TSF with $\tau = 96$

Following that, we transition to a different AR process, allow the models under evaluation to adapt for 200 time steps, and then record their performance over the next 20 episodes before switching tasks again. Figure 4 shows the predictions of different online learning models and TSF-HD after conducting the described experiment. We begin with the warmup process $AR1$ before switching to another process $AR2$ (the results of which are shown in Fig. 4(a), then $AR1$ again (Fig. 4(b), then $AR2$ again (Fig. 4(c), then $AR3$ (Fig. 4(d)), a new process. Further details on each autoregressive process are discussed in Appendix B.1. In this experiment, we focused on short-term distribution shift adaptation, with a forecast horizon of $\tau = 1$. Both the AR-HDC and Seq2Seq-HDC frameworks are identical for $\tau = 1$, and are represented in Figure 4 as 'TSF-HD'. The area around each curve represents one standard deviation around the mean value after repeating the experiment five times. The baselines show more variation and error than TSF-HD, indicating that the linearity of the nonlinear processes in high dimensions enables fast adaptation.

4.2.4. Online Learning Power and Latency Efficiency

We assessed the prediction latency and power usage of TSF-HD and selected online learning baselines on two edge platforms. Our findings for latency and power on two datasets on a Raspberry Pi are detailed in Table 3. Similar results are showcased for the NVIDIA Jetson Nano and for further datasets on the Raspberry Pi in Appendix F. For short-term TSF ($\tau = 3$), AR-HDC demonstrates the lowest latency, followed by Seq2Seq-HDC. Notably, both models maintain a nearly constant latency with minimal standard deviation. For long-term TSF, AR-HDC model falls short, exhibiting the highest latency and reduced power efficiency. This is attributed to its use of a loop for prediction and update phases, updating the model $\tau \gg 1$ times during the update loop. In contrast, Seq2Seq-HDC outperforms the others in both inference latency and power efficiency.

4.2.5. Effect of Dimensionality Reduction

Table 4 illustrates the influence of hypervector dimension D on the Relative Squared Error (RSE) of both Seq2Seq-HDC and AR-HDC models for the ETTh2 and ETTm2 datasets with $\tau = 96$. For Seq2Seq-HDC, the RSE decreases by 85% and 79% for ETTh2 and ETTm2 respectively, as D ranges from 500 to 10k, highlighting the advantages of operating in a high-dimensional space. In contrast, an increase in D does not yield similar benefits for AR-HDC due to the greater data efficiency of the autoregressive formulation.

5. Conclusion

In this work, we present *TSF-HD*, a novel online variable-horizon hyperdimensional time-series prediction framework. By reframing the time-series prediction problem as task-free, online hyperdimensional regression, we exploit the linearity of the nonlinear time series in high dimensions, achieving results superior to the state-of-the-art with higher efficiency.

Acknowledgement

Acknowledgement removed for review.

Impact Statement

This work provides low-overhead solution for edge-based time-series forecasting, potentially enabling consumers and end users to deploy a machine learning forecaster on cheap, rugged hardware with reduced environmental impact from power consumption. While we acknowledge the potential harms from such work, we also note that this provides environmental, economic and accessibility benefits.

One notable concern is the use of more efficient time-series forecasting models for military applications or for unethical financial practices based on information asymmetry. Mitigation strategies to address this may involve increased oversight and monitoring.

Regarding fairness considerations, we believe that the more explainable nature of the hyperdimensional linear regressor allows more thorough research into its bias and fairness. We also note that making sophisticated forecasting capabilities available on cheap hardware and low power platforms contributes to democratizing these capabilities and allowing their use without the expense of complex hardware and high energy use.

References

Aljundi, R., Kelchtermans, K., and Tuytelaars, T. Task-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11254–11263, 2019.

Arani, E., Sarfraz, F., and Zonooz, B. Learning fast, learning slow: A general continual learning method based on complementary learning system. *arXiv preprint arXiv:2201.12604*, 2022.

Bhatnagar, A., Kassianik, P., Liu, C., Lan, T., Yang, W., Cassius, R., Sahoo, D., Arpit, D., Subramanian, S., Woo, G., Saha, A., Jagota, A. K., Gopalakrishnan, G., Singh, M., Krithika, K. C., Maddineni, S., Cho, D., Zong, B., Zhou, Y., Xiong, C., Savarese, S., Hoi, S. C. H., and Wang, H. Merlion: A machine learning library for time series. *CoRR*, abs/2109.09265, 2021. URL https://arxiv.org/abs/2109.09265.

Box, G. E. and Jenkins, G. M. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 17(2):91–109, 1968.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H., and Ranzato, M. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Chen, H., Najafi, M. H., Sadredini, E., and Imani, M. Full stack parallel online hyperdimensional regression on fpga. In *2022 IEEE 40th International Conference on Computer Design (ICCD)*, pp. 517–524. IEEE, 2022.

Dutta, A., Gupta, S., Khaleghi, B., Chandrasekaran, R., Xu, W., and Rosing, T. Hdnn-pim: Efficient in memory design of hyperdimensional computing with feature extraction. In *Proceedings of the Great Lakes Symposium on VLSI 2022*, pp. 281–286, 2022.

Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Grossberg, S. and Grossberg, S. How does a brain build a cognitive code? *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*, pp. 1–52, 1982.

Hernández-Cano, A., Zhuo, C., Yin, X., and Imani, M. Reghd: Robust and efficient regression in hyperdimensional learning system. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 7–12. IEEE, 2021.

Holt, C. C. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20(1):5–10, 2004.

Hyndman, R. and Athanasopoulos, G. *Forecasting: Principles and Practice*. OTexts, Australia, 2nd edition, 2018.

Imani, M., Huang, C., Kong, D., and Rosing, T. Hierarchical hyperdimensional computing for energy efficient classification. In *Proceedings of the 55th Annual Design Automation Conference*, pp. 1–6, 2018.

Kanerva, P. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive computation*, 1: 139–159, 2009.

Kumaran, D., Hassabis, D., and McClelland, J. L. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7):512–534, 2016.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 95–104, 2018.

Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8:293–321, 1992.

Liu, M., Zeng, A., Chen, M., Xu, Z., Lai, Q., Ma, L., and Xu, Q. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35:5816–5828, 2022.

Lopez-Paz, D. and Ranzato, M. A. Gradient episodic memory for continual learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/f87522788a2be2d171666752f97ddebb-Paper.pdf.

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

Pham, Q., Liu, C., Sahoo, D., and Steven, H. Contextual transformation networks for online continual learning. In *International Conference on Learning Representations*, 2020.

Pham, Q., Liu, C., Sahoo, D., and Hoi, S. C. Learning fast and slow for online time series forecasting. *arXiv preprint arXiv:2202.11672*, 2022.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31, 2018.

Räsänen, O. J. Generating hyperdimensional distributed representations from continuous-valued multivariate sensory input. *Cognitive Science*, 2015. URL https://api.semanticscholar.org/CorpusID:15600360.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., and Wayne, G. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Triebe, O., Laptev, N., and Rajagopal, R. Ar-net: A simple auto-regressive neural network for time-series. *arXiv preprint arXiv:1911.12436*, 2019.

Woo, G., Liu, C., Sahoo, D., Kumar, A., and Hoi, S. Cost: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. *arXiv preprint arXiv:2202.01575*, 2022.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023.

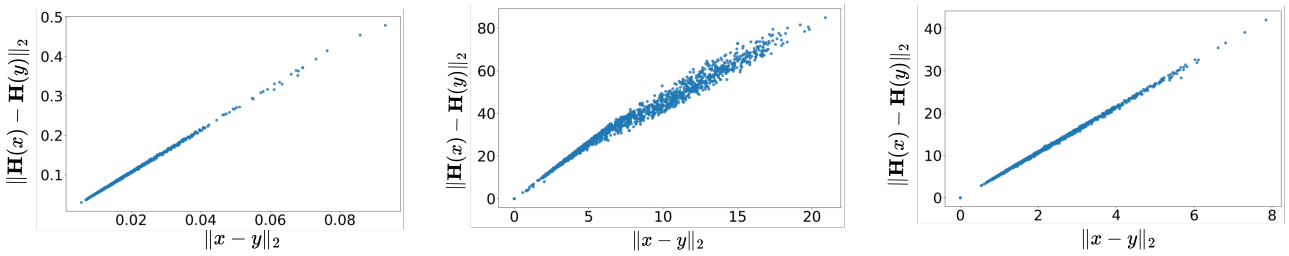
Zhang, Y.-F., Wen, Q., Wang, X., Chen, W., Sun, L., Zhang, Z., Wang, L., Jin, R., and Tan, T. Onenet: Enhancing time series forecasting models under concept drift by online ensembling. *arXiv preprint arXiv:2309.12659*, 2023.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

Appendices

A. Trainable Encoder Validation: Problem Framing

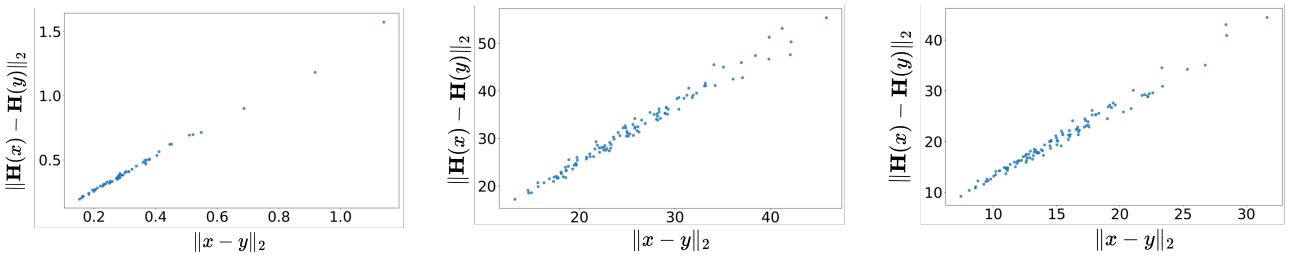
The figure 5 illustrates the distance preservation of the HDC encoder of AR-HDC and Seq2Seq-HDC for a short term TSF ($\tau = 6$) across the Exchange, ETTh1 and the ETTm1 datasets. These readings are similar to the results shown in Figure 1a, with a linear relationship maintained between the distances $\|x - y\|$ separating two points in the data stream (so that $x, y \in \mathcal{X}$) and the encoded distances $\|\mathbf{H}(x) - \mathbf{H}(y)\|$.



(a) *Exchange* (b) *ETTh1* (c) *ETTm1*

Figure 5. Distance preservation across hyperdimensional mappings of input sequences $\mathbf{H}(x)$ for $\tau = 6$

We see a similar observation from Figure 6 regarding distance preservation for long term TSF ($\tau = 96$) on the same datasets. However, we see greater dispersion of points and a greater variation in the linearity of the distance relationship, which may be the cause of the drop in performance for longer horizon forecasting in TSF-HD.



(a) *Exchange* (b) *ETTh1* (c) *ETTm1*

Figure 6. Distance preservation across hyperdimensional mappings of input sequences $\mathbf{H}(x)$ for $\tau = 96$

Figure 7 shows the low average cosine similarity (≤ 0.03) between hypervectors of the encoder matrix W_e, illustrating the orthogonality between $\mathbf{H}$ matrix components for short term forecasting horizon, $\tau = 6$. This illustrates the fact that each component of the encoder is mapping to a different component of the hyperspace, taking advantage of the high dimensionality of the hyperspace compared to the input data.

Similar results are observed for hypervector orthogonality in long term TSF $\tau = 96$ (see Figure 8). However, a similar trend to that between Figures 5 and 6 is also seen, with a higher value of cosine similarity (while still low, at less than 0.05), indicating less efficient use of data thanks to the higher dimensionality of the inputs (larger lookback window).

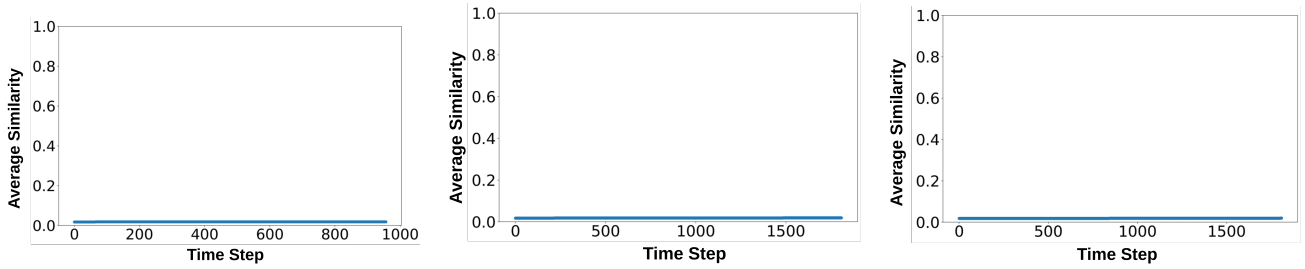
B. Datasets and Evaluation Metrics

B.1. Dataset Details

Details of the real-world benchmark datasets we used are below, with number of variables (dimension of each sample vector), number of time-steps and granularity of sampling in Table 5:

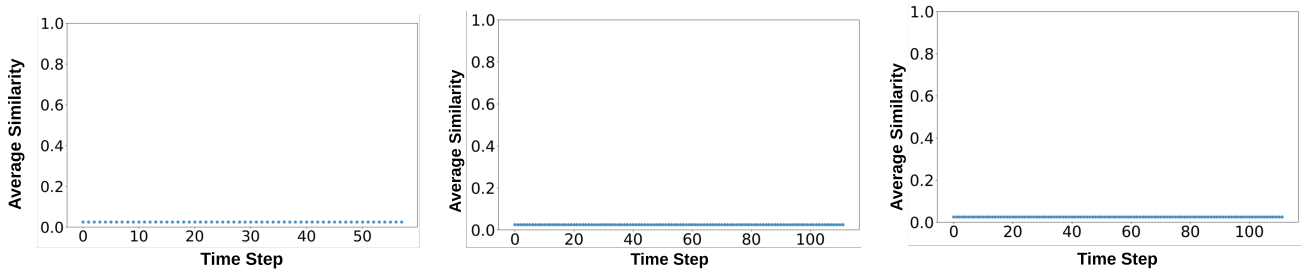
- **ETT**(Zhou et al., 2021)[1] The dataset is composed of two parts: ETTh, which includes hourly data, and ETTm, which

[1] https://github.com/zhouhaoyi/ETDataset



(a) *Exchange* (b) *ETTh1* (c) *ETTm1*

Figure 7. The mean cosine similarity between rows of the encoder matrix W_e of $\mathbf{H}(x)$ used to map low-dimensional input sequences to the hyperdimensional space for $\tau = 6$



(a) *Exchange* (b) *ETTh1* (c) *ETTm1*

Figure 8. The mean cosine similarity between rows of the encoder matrix W_e of $\mathbf{H}(x)$ used to map low-dimensional input sequences to the hyperdimensional space for $\tau = 96$

Datasets	ETTh1 & ETTh2	ETTm1 & ETTm2	ECL	WTH	Exchange	ILI
Variates	7	7	321	12	8	7
Timesteps	17,420	69,680	26,304	52,696	7,588	966
Granularity	1hour	5min	1hour	10min	1day	1week

Table 5. The overall information of the 8 popular TSF Datasets

features data at 15-minute intervals. Each dataset provides information on seven different features related to oil and load in electricity transformers. This data spans a period from July 2016 to July 2018.

- **WTH**[2] The dataset encompasses 21 weather-related metrics, including air temperature and humidity, meticulously recorded every 10 minutes throughout the year 2020 in Germany.
- **Exchange**[3] collects the daily exchange rates of 8 countries from 1990 to 2016.
- **ILI**[4] his dataset details the proportion of patients presenting with influenza-like illness compared to the total number of patients seen. It comprises weekly records from the U.S. Centers for Disease Control and Prevention, spanning from 2002 to 2021
- **ECL**[5] collects the hourly electricity consumption of 321 clients from 2012 to 2014.

We also use the Synthetic-Abrupt univariate time series **S-A** dataset, drawing from (Pham et al., 2022). It is a concatenation of first-order auto-regressive processes $AR_{\varphi}(\sigma)$ defined as

$$X_t = \varphi X_{t-1} + \epsilon_t, \tag{7}$$

[2] https://www.bgc-jena.mpg.de/wetter/
[3] https://github.com/laiguokun/multivariate-time-series-data
[4] https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html
[5] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

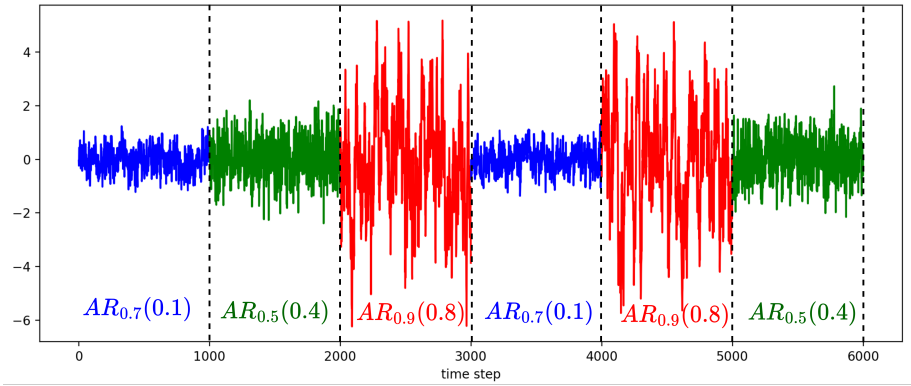


Figure 9. Overview of Synthetic Abrupt time series (S-A)

where ϵ_t is zero-mean Gaussian noise with variance σ. The first term X_0 is randomly generated from a zero-mean Gaussian distribution. The S-Abrupt dataset is generated with different processes acting at different time intervals, as described by the following equation:

$$X_t = \begin{cases} AR1 = AR_{0.7}(0.1) & \text{if } 1 < t \leq 1000 \\ AR3 = AR_{0.5}(0.4) & \text{if } 1000 < t \leq 1999 \\ AR2 = AR_{0.9}(0.8) & \text{if } 2000 < t \leq 2999 \\ AR1 = AR_{0.7}(0.1) & \text{if } 3000 < t \leq 3999 \\ AR2 = AR_{0.9}(0.8) & \text{if } 4000 < t \leq 4999 \\ AR3 = AR_{0.5}(0.4) & \text{if } 5000 < t \leq 5999. \end{cases} \tag{8}$$

The **S-A** implementation in this paper is different from the one proposed in (Pham et al., 2022), as we have changed the variance of ϵ_t in each of the AR processes to accentuate the severity of the task shifts (concept drift) in the time series. Earlier work (Pham et al., 2022) retained identical noise variance for all AR processes.

Figure 9 illustrates the S-A dataset over time with a color associated to each AR process. It begins with a warmup process $AR1$ followed by $AR2$, $AR1$ again, $AR2$ again and $AR3$, to evaluate forecaster adaptation to task shifts *and* recurrence of old tasks. Figure 4 of Section 4.2.3 illustrates concept drift adaptation for TSF-HD (main body of the paper) across this dataset. Section 4.2.3 skips the performance analysis for the first appearance of $AR3$ for brevity, but we run the online learning systems across the entire dataset shown in Figure 9. We allow online learning to occur for the first 200 of the 1000 timesteps of each process, followed by comparing predictions and learning online to state of the art for the next 20 time steps (following the setup of (Pham et al., 2022)).

B.2. Metrics

The metrics used to evaluate forecaster accuracy are $CORR$ (Empirical Correlation Coefficient) (Lai et al., 2018) and RSE (Relative Root Squared Error) (Lai et al., 2018) detailed here:

$$RSE(\tilde{X}, X) = \frac{\sqrt{\sum_{i=0}^{\tau}(\tilde{x}_i - x_i)^2}}{\sqrt{\sum_{i=0}^{\tau}(x_i - \bar{X})^2}} \tag{9}$$

$$CORR(\tilde{X}, X) = \frac{1}{d}\sum_{j=0}^{d} \frac{\sum_{i=0}^{\tau}(x_{i,j} - \bar{X}_j)(\tilde{x}_{i,j} - \bar{\tilde{X}}_j)}{\sum_{i=0}^{\tau}(x_{i,j} - \bar{X}_j)^2(\tilde{x}_{i,j} - \bar{\tilde{X}}_j)^2} \tag{10}$$

in the RSE and $CORR$ equations, $\tilde{x}$ and x refer respectively to the predicted sample and the ground truth sample. $\bar{\tilde{X}}$ refers to the mean of the entire predicted sequence while $\bar{X}$ refers to the mean of the entire ground truth sequence. For RSE, a lower value indicates a lower average squared error when compared to the naive forecaster (forecasting all values as the sequence mean), and thus indicates better accuracy. For CORR, the correlation coefficient ranges from 0 to 1 and indicates how well-correlated the forecaster predictions are with the data stream - a higher value indicates more correlated predictions (better precision).

C. Baselines

		Seq2Seq-HDC		AR-HDC		SCINet		Nlinear		Naive		GBRT	
	τ	*RSE*	*CORR*	*RSE*	*CORR*	*RSE*	*CORR*	*RSE*	*CORR*	*RSE*	*CORR*	*RSE*	*CORR*
Exchange	3	0.032	0.999	0.033	0.999	0.053	0.999	0.017	0.999	**0.011**	**0.999**	0.018	0.999
	6	0.043	0.999	0.034	0.999	0.052	0.999	0.021	0.999	**0.017**	**0.999**	0.021	0.999
	12	0.053	0.998	0.035	0.999	0.05	0.999	0.028	0.999	**0.023**	**0.999**	0.027	0.999
ECL	3	0.314	0.952	0.288	**0.966**	**0.129**	0.902	0.331	0.945	0.329	0.945	0.381	0.925
	6	**0.153**	**0.988**	0.305	0.976	0.154	0.875	0.503	0.873	0.493	0.879	0.47	0.882
	12	**0.106**	**0.994**	0.121	**0.994**	0.144	0.866	0.732	0.731	0.733	0.739	0.596	0.809
ETTh2	3	0.142	0.989	0.128	**0.994**	0.179	0.984	**0.124**	0.992	0.125	0.992	0.134	0.991
	6	0.162	0.987	0.177	**0.991**	0.209	0.982	**0.147**	0.989	0.174	0.985	0.167	0.986
	12	0.178	0.984	0.172	**0.988**	0.21	0.978	**0.169**	0.985	0.234	0.973	0.207	0.978
ETTh1	3	0.389	**0.928**	**0.349**	0.92	0.406	0.916	0.396	0.917	0.562	0.842	0.623	0.785
	6	0.482	**0.894**	**0.443**	0.875	0.468	0.885	0.47	0.882	0.892	0.609	0.817	0.592
	12	**0.371**	**0.938**	0.448	0.902	0.53	0.86	0.517	0.855	1.227	0.289	1.011	0.362
ETTm1	3	0.111	**0.994**	**0.11**	**0.994**	0.302	0.957	0.253	0.968	0.267	0.964	0.422	0.914
	6	**0.135**	**0.991**	0.148	0.99	0.411	0.916	0.338	0.943	0.363	0.934	0.484	0.879
	12	**0.181**	**0.984**	0.209	0.982	0.517	0.86	0.503	0.874	0.538	0.854	0.609	0.793
ETTm2	3	0.104	0.994	**0.086**	**0.998**	0.162	0.989	0.081	0.996	0.087	0.996	0.102	0.994
	6	0.136	0.991	0.114	**0.996**	0.134	0.991	0.097	0.995	**0.103**	0.995	0.113	0.993
	12	0.161	0.987	0.171	**0.993**	0.174	0.984	0.121	0.992	**0.127**	0.992	0.133	0.991
WTH	3	0.671	0.784	0.642	0.797	0.636	0.78	0.724	0.737	0.711	0.742	**0.596**	**0.803**
	6	0.712	0.773	0.674	**0.785**	**0.647**	0.773	0.797	0.68	0.788	0.681	0.65	0.763
	12	0.665	0.799	0.651	**0.802**	**0.645**	0.77	0.854	0.633	0.834	0.635	0.68	0.737
ILI	3	0.203	0.979	0.135	**0.998**	0.23	0.983	0.118	0.993	0.103	0.996	**0.093**	0.996
	6	0.247	0.969	0.202	**0.995**	0.364	0.948	**0.153**	0.988	0.169	0.99	0.155	0.991
	12	0.296	0.956	0.297	**0.989**	0.383	0.953	0.196	0.981	0.252	0.982	**0.236**	0.985

Table 6. Short Term Time Series Forecasting Performance of AR-HDC & Seq2Seq-HDC compared to other offline baselines. We report the mean of RSE and CORR of the experiments. The results in bold are the best and in blue and underlined are second best

The details of the baselines we have evaluated TSF-HD against are shown below:

- **OnlineTCN**[6] uses a standard TCN backbone (Woo et al., 2022) with 10 hidden layers, each of which has two stacks of residual convolution filters.
- **ER**[6] (Chaudhry et al., 2019) ER enhances OnlineTCN by adding episodic memory to mix old and new learning samples.
- **DER++**[6](Buzzega et al., 2020) augments the standard ER with a ℓ_2 knowledge distillation loss on the previous logits.
- **FSNet**[6] (Pham et al., 2022) or Fast and Slow learning Network is an online learning technique combining rapid adaptation to new data and memory recall of past events.
- **Informer**[7] (Zhou et al., 2021) is a transformer-based model employing self-attention for efficiency and a generative decoder for accurate predictions.
- **SCINet**[8](Liu et al., 2022) is an architecture that enhances TSF by recursively downsampling and convolving data to extract complex temporal features.
- **NLinear**[9] It enhances LTSF-Linear's (Zeng et al., 2023) performance on shifting datasets by normalizing inputs through subtraction and addition around a linear layer.
- **Naive**[9] is a naive direct multi-step which repeats the last value in the look-back window.
- **GBRT**[9] is the classical Gradient Boosting Regression Trees algorithm (Friedman, 2001).

The online learning and transformer baselines' evaluation against TSFHD are shown in Section 4.2.1, while the results of the offline learning and classical algorithms' comparison to TSF-HD are shown in Appendix D.

[6] https://github.com/salesforce/fsnet
[7] https://github.com/zhouhaoyi/Informer2020
[8] https://github.com/cure-lab/SCINet
[9] https://github.com/cure-lab/LTSF-Linear

		Seq2Seq-HDC		AR-HDC		SCINet		Nlinear		Naive		GBRT	
	tau	***RSE***	***CORR***	***RSE***	***CORR***	***RSE***	***CORR***	***RSE***	***CORR***	***RSE***	***CORR***	***RSE***	***CORR***
ETTh1	96	0.601	**0.846**	**0.568**	0.84	0.698	0.753	0.643	0.7739	1.208	0.273	1.01	0.347
	192	0.678	0.803	**0.599**	**0.827**	0.763	0.7	0.671	0.752	1.226	0.255	1.02	0.328
	384	0.797	0.739	**0.628**	**0.831**	0.864	0.66	0.687	0.735	1.239	0.247	1.03	0.321
ETTm1	96	0.315	0.951	**0.297**	**0.965**	0.585	0.817	0.587	0.813	1.127	0.345	0.985	0.405
	192	0.392	0.925	**0.373**	**0.949**	0.672	0.767	0.658	0.762	1.143	0.323	1.01	0.375
	384	0.465	0.893	**0.431**	**0.924**	1.064	0.602	0.731	0.698	1.166	0.296	1.033	0.343
ILI	24	0.288	0.957	0.229	**0.986**	0.382	0.934	**0.173**	0.985	0.304	0.98	0.277	0.983
	36	0.398	0.918	0.199	**0.99**	0.42	0.969	**0.176**	0.985	0.334	0.981	0.307	0.984
	48	0.448	0.901	0.204	**0.989**	0.481	0.965	**0.185**	0.984	0.33	0.968	0.303	0.972
Exchange	96	0.211	0.978	0.097	0.997	0.092	0.996	**0.062**	**0.998**	0.062	**0.998**	0.062	**0.998**
	192	*0.318*	0.948	0.155	0.994	0.159	0.989	0.089	**0.996**	**0.088**	**0.996**	0.088	**0.996**
	224	0.328	0.945	0.137	0.995	0.17	0.987	0.096	**0.996**	0.091	**0.996**	**0.09**	**0.996**
ECL	96	**0.177**	0.985	0.206	**0.989**	0.364	0.933	0.726	0.736	0.722	0.735	0.571	0.823
	192	**0.214**	0.977	0.294	**0.979**	0.551	0.857	0.731	0.733	0.721	0.736	0.572	0.822
	384	**0.282**	0.962	0.302	**0.979**	0.796	0.631	0.737	0.729	0.729	0.732	0.581	0.817
WTH	96	0.697	0.793	**0.633**	**0.796**	0.651	0.767	0.898	0.593	0.915	0.585	0.746	0.687
	192	0.723	0.784	**0.632**	0.791	0.691	0.744	0.906	**0.856**	0.925	0.575	0.756	0.678
	384	0.777	0.762	0.653	0.776	0.777	0.692	**0.579**	**0.916**	0.938	0.566	0.767	0.669

Table 7. Long Term Time Series Forecasting Performance of AR-HDC & Seq2Seq-HDC compared to the offline baseline. We report the mean of RSE and CORR of the experiments. The results in bold are the best and in blue and underlined are second best

D. Additional results

In Table 6 we compare Seq2Seq-HDC and AR-HDC to convolutional, linear, naive and gradient boosting approaches for short term TSF. Aside from the Exchange Rate dataset, where the naive and the linear models slightly outperform our techniques (a similar observation is seen in (Zeng et al., 2023)), TSF-HD (either AR-HDC or Seq2Seq-HDC) outperforms all offline methods in either the CORR or RSE metrics.

In Table 7, AR-HDC outperforms all offline baselines across most test cases, once again with the exception of the Exchange dataset. In this specific case, the Linear model (NLinear (Zeng et al., 2023)) and the naive approach prove to be more precise than other techniques. This outcome aligns with previous results where the same baseline surpassed our models in short-term TSF, and aligns with prior work.

For the majority of test cases, AR-HDC demonstrates superior performance over Seq2Seq-HDC. Notably, AR-HDC, having fewer parameters, tends to converge more rapidly than Seq2Seq-HDC, particularly for long-term TSF. This faster convergence is likely due to underfitting, especially when there are fewer learning steps (i.e., the number of times the samples are presented) in long-term TSF, as the time series samples should not overlap and TSF-HD is trained periodically on the samples it predicted up to the forecast horizon - the larger the horizon, the less frequently TSF-HD models are trained.

In contrast to the linear model (NLinear) which projects the T-dimensional input space into a τ-dimensional prediction space, AR-HDC and Seq2Seq-HDC first project the T-dimensional space into an hyperspace of D-dimension ($D \gg max(T, \tau)$) then project it back to the τ-dimensional space. This operation extracts further information from the time series and better retains information about old tasks, allowing better precision for almost all the short term and long term TSF cases save for Exchange.

		AR-HDC	Seq2Seq-HDC	FSNet	ER	DER++	OnlineTCN
ECL	3	$0.073^{\pm 0.018}$	$\mathbf{0.037}^{\pm \mathbf{0.021}}$	$0.356^{\pm 0.099}$	$0.364^{\pm 0.031}$	$0.365^{\pm 0.036}$	$0.169^{\pm 0.079}$
	96	$5.233^{\pm 0.669}$	$\mathbf{0.0828}^{\pm \mathbf{0.041}}$	$1.44^{\pm 0.277}$	$1.791^{\pm 0.061}$	$1.791^{\pm 0.065}$	$0.991^{\pm 0.097}$
ETTh1	3	$\mathbf{0.0144}^{\pm \mathbf{0.001}}$	$0.065^{\pm 0.011}$	$0.47^{\pm 0.122}$	$0.33^{\pm 0.024}$	$0.335^{\pm 0.023}$	$0.212^{\pm 0.078}$
	96	$0.867^{\pm 0.03}$	$\mathbf{0.071}^{\pm \mathbf{0.038}}$	$0.530^{\pm 0.209}$	$0.696^{\pm 0.375}$	$0.697^{\pm 0.399}$	$0.295^{\pm 0.125}$
Exchange	3	$\mathbf{0.016}^{\pm \mathbf{0.002}}$	$0.066^{\pm 0.013}$	$0.452^{\pm 0.121}$	$0.334^{\pm 0.024}$	$0.337^{\pm 0.025}$	$0.213^{\pm 0.079}$
	96	$0.937^{\pm 0.347}$	$\mathbf{0.073}^{\pm \mathbf{0.041}}$	$0.538^{\pm 0.227}$	$0.702^{\pm 0.036}$	$0.701^{\pm 0.039}$	$0.273^{\pm 0.093}$
WTH	3	$\mathbf{0.015}^{\pm \mathbf{0.003}}$	$0.069^{\pm 0.004}$	$0.329^{\pm 0.09}$	$0.304^{\pm 0.035}$	$0.305^{\pm 0.308}$	$0.149^{\pm 0.082}$
	96	$1.171^{\pm 0.654}$	$\mathbf{0.082}^{\pm \mathbf{0.038}}$	$0.678^{\pm 0.316}$	$0.730^{\pm 0.043}$	$0.723^{\pm 0.046}$	$0.305^{\pm 0.117}$

Table 8. Latency of Online learning models on RaspberryPI ($mean^{\pm std}$)

		AR-HDC	Seq2Seq-HDC	FSNet	ER	DER++	OnlineTCN
ECL	3	$4.715^{\pm 0.976}$	$\mathbf{4.153^{\pm 0.932}}$	$4.827^{\pm 0.742}$	$4.881^{\pm 0.746}$	$4.914^{\pm 0.739}$	$4.471^{\pm 0.817}$
	96	$5.548^{\pm 0.506}$	$\mathbf{4.762^{\pm 0.816}}$	$4.923^{\pm 0.582}$	$5.07^{\pm 0.646}$	$5.077^{\pm 0.649}$	$4.914^{\pm 0.552}$
ETTh1	3	$\mathbf{3.508^{\pm 0.486}}$	$4.236^{\pm 0.904}$	$4.766^{\pm 0.637}$	$4.833^{\pm 0.795}$	$4.906^{\pm 0.788}$	$4.567^{\pm 0.783}$
	96	$5.286^{\pm 0.203}$	$\mathbf{4.419^{\pm 0.967}}$	$4.961^{\pm 0.591}$	$5.194^{\pm 0.753}$	$5.247^{\pm 0.743}$	$4.858^{\pm 0.698}$
Exchange	3	$\mathbf{3.395^{\pm 0.534}}$	$4.246^{\pm 0.945}$	$4.855^{\pm 0.635}$	$4.974^{\pm 0.761}$	$4.982^{\pm 0.732}$	$4.654^{\pm 0.748}$
	96	$5.257^{\pm 0.511}$	$\mathbf{4.437^{\pm 0.855}}$	$4.975^{\pm 0.642}$	$5.244^{\pm 0.732}$	$5.259^{\pm 0.742}$	$4.788^{\pm 0.804}$
WTH	3	$\mathbf{3.457^{\pm 0.469}}$	$3.971^{\pm 1.122}$	$4.752^{\pm 0.788}$	$4.903^{\pm 0.791}$	$4.804^{\pm 0.901}$	$4.338^{\pm 0.919}$
	96	$5.272^{\pm 0.429}$	$\mathbf{4.450^{\pm 0.87}}$	$4.931^{\pm 0.679}$	$5.255^{\pm 0.716}$	$5.208^{\pm 0.768}$	$4.818^{\pm 0.725}$

Table 9. Power consumption of Online learning models on RaspberryPI ($mean^{\pm std}$)

		AR-HDC	Seq2Seq-HDC	FSNet	ER	DER++	OnlineTCN
ECL	3	$0.035^{\pm 0.008}$	$\mathbf{0.018^{\pm 0.008}}$	$0.637^{\pm 1.638}$	$0.229^{\pm 0.662}$	$0.286^{\pm 1.167}$	$0.193^{\pm 0.847}$
	96	$1.024^{\pm 0.074}$	$\mathbf{0.022^{\pm 0.02}}$	$0.662^{\pm 1.403}$	$0.403^{\pm 1.384}$	$0.409^{\pm 1.524}$	$0.295^{\pm 1.456}$
ETTh1	3	$0.051^{\pm 0.187}$	$\mathbf{0.031^{\pm 0.151}}$	$0.627^{\pm 1.341}$	$0.283^{\pm 1.194}$	$0.239^{\pm 0.731}$	$0.182^{\pm 0.702}$
	96	$0.923^{\pm 0.043}$	$\mathbf{0.018^{\pm 0.008}}$	$0.688^{\pm 1.897}$	$0.264^{\pm 0.919}$	$0.242^{\pm 0.695}$	$0.178^{\pm 0.749}$
Exchange	3	$0.036^{\pm 0.025}$	$\mathbf{0.02^{\pm 0.017}}$	$0.644^{\pm 1.408}$	$0.231^{\pm 0.641}$	$0.235^{\pm 0.628}$	$0.166^{\pm 0.621}$
	96	$0.938^{\pm 0.065}$	$\mathbf{0.021^{\pm 0.017}}$	$0.622^{\pm 1.209}$	$0.238^{\pm 0.608}$	$0.241^{\pm 0.649}$	$0.169^{\pm 0.642}$
WTH	3	$0.035^{\pm 0.024}$	$\mathbf{0.017^{\pm 0.010}}$	$0.631^{\pm 1.214}$	$0.235^{\pm 0.703}$	$0.233^{\pm 0.666}$	$0.176^{\pm 0.706}$
	96	$0.927^{\pm 0.049}$	$\mathbf{0.017^{\pm 0.007}}$	$0.661^{\pm 1.66}$	$0.244^{\pm 0.697}$	$0.249^{\pm 0.701}$	$0.176^{\pm 0.676}$

Table 10. Latency of Online learning models on Nvidia Jetson Nano($mean^{\pm std}$)

E. Hardware Setup

To evaluate the precision of TSF-HD and compare it against the baselines (as in Section 4.2.1 and Appendix D), we trained our models (AR-HD & Seq2Seq-HD) on an Nvidia RTX 3050 with 4GB of RAM. The baseline models except for GBRT (Friedman, 2001) were trained on an Nvidia RTX A2000 with 12GB of RAM. GBRT (Friedman, 2001) was trained on a CPU 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz.

To evaluate the inference latency and power overhead of TSF-HD in comparison to the baselines, we conducted measurements on a RaspberryPi4 (Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8GHz) and on an Nvidia-Jetson Nano with 4 GB of RAM. The power usage of the RaspberryPi4 was measured with a USB Powermeter placed between the board and the Power Supply Unit (PSU). For the Nvidia-Jetson the power was measured internally using the python library `Jtop`. We note that some baselines such as Informer (Zhou et al., 2021) are unable to fit on a Raspberry Pi and therefore were not considered for inference latency and power measurement.

F. Latency and Power Results

Our complete findings for latency and power on a Raspberry Pi are detailed in Table 8 (latency) and Table 9 (power) for four different datasets and two different values of τ representing short and long-term forecast horizons, comparing the two TSF-HD models with our online learning baselines. For short-term TSF ($\tau = 3$), AR-HDC demonstrates the lowest latency, followed by Seq2Seq-HDC. Notably, both models maintain a nearly constant latency with minimal standard deviation. For long-term TSF, AR-HDC falls short, exhibiting the highest latency and reduced power efficiency. This is attributed to its use of a loop for prediction and update phases, updating the model $\tau \gg 1$ times during the update loop. In contrast, Seq2Seq-HDC outperforms the others in both inference latency and power efficiency thanks to its use of one-shot prediction. Seq2Seq-HDC is seen to be faster than baseline models and AR-HDC on edge GPUs for both short and long-term TSF. On edge CPUs, AR-HDC is faster and more power-efficient the other models, but only for short-term forecasting scenarios.

Tables 10 and 11 present similar results, comparing latency and power consumption of AR-HDC and Seq2Seq-HDC with the online learning baselines over four datasets and two values of τ for short- and long-term TSF on an Nvidia Jetson Nano edge GPU. Unlike the ARM CPU of the Raspberry Pi, the edge GPU is better suited for parallel computing, enabling faster matrix computations. This advantage is evident as Seq2Seq-HDC outpaces AR-HDC in short-term TSF as well as long-term TSF on this platform. From a power consumption standpoint, AR-HDC and Seq2Seq-HDC rank as the most and second-most efficient models, respectively, except in the Exchange dataset for short-term forecasting (where Online

		AR-HDC	Seq2Seq-HDC	FSNet	ER	DER++	OnlineTCN
ECL	3	$3.58^{\pm 0.257}$	$\mathbf{3.296^{\pm 0.164}}$	$4.021^{\pm 0.465}$	$3.362^{\pm 0.121}$	$3.344^{\pm 0.097}$	$3.363^{\pm 0.264}$
	96	$5.501^{\pm 0.501}$	$\mathbf{3.755^{\pm 0.429}}$	$4.141^{\pm 0.397}$	$5.222^{\pm 0.526}$	$5.17^{\pm 0.519}$	$5.047^{\pm 0.378}$
ETTh1	3	$\mathbf{3.184^{\pm 0.109}}$	$3.3^{\pm 0.017}$	$4.188^{\pm 0.303}$	$3.318^{\pm 0.129}$	$3.234^{\pm 0.056}$	$3.302^{\pm 0.211}$
	96	$3.523^{\pm 0.086}$	$\mathbf{3.393^{\pm 0.287}}$	$4.234^{\pm 0.316}$	$4.307^{\pm 0.621}$	$4.688^{\pm 0.524}$	$3.636^{\pm 0.179}$
Exchange	3	$\mathbf{3.395^{\pm 0.211}}$	$3.532^{\pm 0.142}$	$4.04^{\pm 0.3}$	$3.417^{\pm 0.126}$	$3.487^{\pm 0.29}$	$3.262^{\pm 0.134}$
	96	$\mathbf{3.539^{\pm 0.089}}$	$3.706^{\pm 0.318}$	$4.289^{\pm 0.351}$	$4.474^{\pm 0.596}$	$4.337^{\pm 0.449}$	$3.73^{\pm 0.043}$
WTH	3	$3.157^{\pm 0.02}$	$\mathbf{3.118^{\pm 0.125}}$	$4.148^{\pm 0.309}$	$3.298^{\pm 0.06}$	$3.362^{\pm 0.155}$	$3.288^{\pm 0.114}$
	96	$3.536^{\pm 0.088}$	$3.502^{\pm 0.406}$	$4.265^{\pm 0.291}$	$4.394^{\pm 0.545}$	$4.243^{\pm 0.423}$	$\mathbf{3.451^{\pm 0.099}}$

Table 11. Power consumption of Online learning models on Nvidia Jetson Nano ($mean^{\pm std}$)

TCN is the second most power-efficient) and Weather dataset for long-term forecasting (where OnlineTCN is the most power-efficient). To summarize, the Seq2Seq-HDC model proves more efficient than baseline models and AR-HDC on edge GPUs for both short and long-term TSF. Conversely, on edge CPUs, AR-HDC demonstrates lower overhead than Seq2Seq-HDC and other baseline models, but only in short-term forecasting scenarios.

G. Reproducibility Details

The experiments are performed using 5 different random seed values, which are 2019, 2020, 2021, 2022 and 2023. The learning rate used are summarized in the table 12. We note that the experiments can be replicated using the provided GitHub repository.

	Exchange	ECL	ETTh1	ETTh2	ETTm1	ETTm2	WTH	ILI
Seq2Seq-HDC	1e-3	2e-5	1e-4	1e-4	1e-4	1e-4	1e-4	1e-3
AR-HDC	1e-4	4e-5	5e-5	5e-5	5e-5	5e-5	4e-4	1e-3

Table 12. Learning rate parameter values