MATS: MEMORY ATTENTION FOR TIME-SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-term time series forecasting (LTSF) is still very challenging in many real-world applications. A fundamental difficulty is in efficiently modeling both the short-term temporal patterns and long-term dependencies. In this paper, we introduce a novel two-stage attention-based LTSF model called **M**emory **A**ttention for **T**ime-**S**eries forecasting (MATS). In stage I, short-term temporal patterns are extracted to a memory bank such that the input time series is represented by a much shorter sequence of memory attentions. In stage II, a sequence-to-sequence predictor is trained to discover long-term dependencies in the memory attention sequence, and forecast memory attentions corresponding to the time series in the future. The use of attention allows a flexible representation, and its shorter sequence length enables the model to more easily learn long-term dependencies. Extensive experiments on a number of multivariate and univariate benchmark datasets demonstrate that MATS outperforms SOTA LTSF methods almost all the time.

1 INTRODUCTION

Long-term time series forecasting (LTSF) has recently attracted a lot of attention in various domains such as sensor network monitoring (Papadimitriou & Yu, 2006), energy consumption (Deb et al., 2017), traffic and economics planning (Zhu & Shasha, 2002), and weather forecasting (Matsubara et al., 2014). Obviously, LTSF is more challenging than traditional short-term prediction (e.g., one-step-ahead prediction) as it requires forecasting longer into the future. Popular LTSF models are based on transformers that learn long-range dependencies using multi-head self-attention. Examples include LogTrans (Li et al., 2019), Reformer (Kitaev et al., 2020), Informer (Zhou et al., 2021), Pyraformer (Liu et al., 2021b), Autoformer (Wu et al., 2021), and FEDformer (Zhou et al., 2022). Although these transformer variants improve LTSF by benefiting from the self-attention mechanism to capture long-term dependencies, they are still limited in modeling very long time series and capturing the local short-term context. Without learning and modeling the dependencies among an abundant number of local patterns, generating an accurate long-term prediction is challenging.

DLinear (Zeng et al., 2022) is a recent model that often outperforms transformer models in LTSF. Inspired by the Autoformer, it is also based on the season-trend time series decomposition, but uses shallow linear layers for modeling temporal dependencies. However, these linear layers may not be sufficient for complicated real-world time series. Moreover, short-term patterns are extracted only from the lookback window but not from the whole time series, limiting the model's representation ability.

In computer vision, it has been observed that local patterns and global dependencies are both critical in processing high-resolution images. VQ-VAE (Van D. et al., 2017) and VQGAN (Esser et al., 2021) address this by using a two-stage training procedure. The first stage focuses on using the convolutional neural network (CNN) to extract local patterns, which are then stored in a vector quantization (VQ) codebook. The second stage focuses on capturing global dependencies among codebook entries and forecasting new codebook indices by auto-regression given the past codebook indices. This two-stage approach allows VQ-VAE and VQGAN to learn local patterns and global dependencies separately and more easily without interfering each other.

Motivated by VQ-VAE and VQGAN, in this paper, we introduce a two-stage LTSF model called **M**emory **A**ttention for **T**ime-**S**eries forecasting (MATS). Instead of using a VQ codebook as in VQ-VAE and VQGAN, we propose to use an auto-encoder and a memory bank to store abundant local patterns in the memory units. At each time step, the time series is represented as a combination of these local patterns, weighted by the attention scores between the time series and individual memory units. While the VQ representation can only use one codebook entry each time, attention allows a more powerful representation of the local temporal patterns, as supported by the success of transformers in various data modalities. The auto-encoder and memory bank are learned together in stage I and then frozen. In stage II, a sequence-to-sequence predictor is trained to discover long-term dependencies in the memory attention sequence. On forecasting, a new memory attention subsequence corresponding to the time series in the future is output from the learned sequence-to-sequence predictor, which is then decoded by the decoder (in the auto-encoder) to generate the time series forecast.

Our contributions can be summarized as follows: (i) We develope a novel two-stage model for LTSF; (MATS); (ii) We propose using the memory attention sequence as a flexible representation of the local temporal patterns and a sequence-to-sequence predictor to extract long-term dependencies in the memory attention sequence; (iii) We perform comprehensive experiments and comparisons with existing state-of-the-art LTSF baselines, and demonstrate that the proposed model achieves much better prediction performance.

2 Related Work

Traditional time series forecasting methods are usually based on statistical auto-regressive models (e.g., Autoregressive Integrated Moving Average (ARIMA) (Box & Jenkins, 1968; Box & Pierce, 1970)) and gradient boosting tree (GBRT) (Friedman, 2001; Elsayed et al., 2021). However, they assume simple linear temporal dependencies or rely on manual feature selection.

In the past decade, deep learning has become a powerful class of representation learning methods. Various deep learning solutions, such as the recurrent neural network (RNN) and its variants (Hochreiter & Schmidhuber, 1997; Wen et al., 2017; Rangapuram et al., 2018; Yu et al., 2017; Maddix et al., 2018), have been developed for LTSF. Based on the RNN, DeepAR (Salinas et al., 2020) uses binomial likelihood for sequential probabilistic forecasting. Attention-based RNNs (Wu et al., 2020; Shih et al., 2019; Song et al., 2018; Qin et al., 2017) introduce the attention mechanism in the time dimension to capture long-term dependencies. Besides, convolutional neural networks, though initially used in computer vision, are also popular in time series modeling (van den Oord et al., 2016; Borovykh et al., 2017; Bai et al., 2018). The LSTNet (Lai et al., 2018) introduces CNN to capture short-term temporal patterns. TCN (Sen et al., 2019) models temporal causality with causal convolution. SCINet (Liu et al., 2021a) uses convolutions at multiple temporal resolutions. However, RNN and CNN usually excel only in short-term, not long-range, forecasting.

With the tremendous success of the transformer in natural language processing (Vaswani et al., 2017; Kenton & Toutanova, 2019; Brown et al., 2020), many recent attempts for LTSF are based on the transformer (Li et al., 2019; Kitaev et al., 2020; Liu et al., 2021b). For example, Informer (Zhou et al., 2021) reduces the transformer complexity by using the direct multi-step forecasting objective. Autoformer (Wu et al., 2021) enhances the transformer by auto-correlation attention and season-trend decomposition. Based on the Autoformer, the FEDformer (Zhou et al., 2022) introduces frequency-attention blocks and trend components extracted by various kernel sizes.

Besides the use of RNNs, CNNs, and transformers, there are recent attempts with other neural network models. N-Beats (Oreshkin et al., 2019) uses a feedforward network and neural basis function approximation. Dlinear (Zeng et al., 2022) integrates season-trend decomposition with linear layers. DeepTIMe (Woo et al., 2022) explores a deep time-index-based model using meta-learning. Surprisingly, they often outperform many recent transformer-based models. However, as real-world time series can be long, noisy, and non-stationary, LTSF is still difficult in general.

3 proposed Model

Given a C-variate time series segment $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T]$ (where $\boldsymbol{x}_i \in \mathbb{R}^C$) of length T, the LTSF task is to predict its H future values $\tilde{\boldsymbol{X}} = [\boldsymbol{x}_{T+1}, \boldsymbol{x}_{T+2}, \ldots, \boldsymbol{x}_{T+H}]$. The proposed MATS (shown in Figure 1) involves two stages of operation.

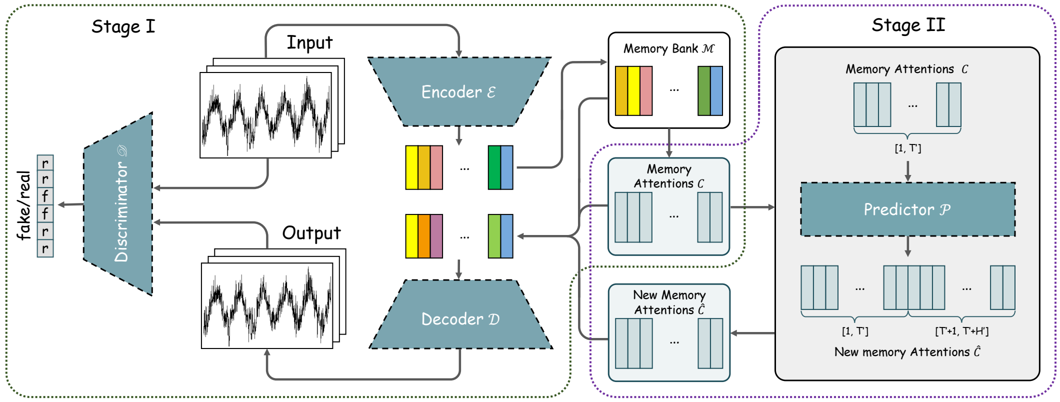


Figure 1: Overview of MATS. The left half part shows the first stage components and the right half part show the second stage components.

In stage I, we train an auto-encoder and a memory bank to extract local patterns. The input time series segment is then represented as a sequence of attention vectors on the memory units. In stage II, both the auto-encoder and memory bank are fixed. A sequence-to-sequence model is used to predict the missing attention vectors corresponding to the future time series segment. Finally, the reconstructed attention sequence is decoded by the decoder (in the auto-encoder) to output the desired time series forecast.

3.1 Stage I

Stage I involves an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$, which encodes/decodes $\boldsymbol{X}$ to/from a sequence of hidden representations. We use an L-layer convolutional neural network (CNN) for $\mathcal{E}$. With strides larger than 1, the output of the last convolution layer is shortened to a length-T' sequence $\mathcal{E}(\boldsymbol{X}) = \boldsymbol{H} = [\boldsymbol{h}_1, \dots, \boldsymbol{h}_{T'}] \in \mathbb{R}^{d \times T'}$, where d is the feature dimensionality. Since we use a CNN, the length T of input $\boldsymbol{X}$ can be arbitrary, and the output length T' will be changed accordingly. As for the decoder $\mathcal{D}$, we use a deconvolution network (Zeiler et al., 2010) which reconstructs $\boldsymbol{X}$ from $\boldsymbol{H}$, as $\hat{\boldsymbol{X}} = \mathcal{D}(\boldsymbol{H})$.

Inspired by Weston et al. (2015), we introduce a memory bank $\mathcal{M}$ to store common local patterns extracted from all the time series segments. $\mathcal{M}$ has M learnable d-dimensional memory units $\boldsymbol{M} = [\boldsymbol{m}_1, \dots, \boldsymbol{m}_M] \in \mathbb{R}^{d \times M}$. For each $\boldsymbol{h}_t$ in $\boldsymbol{H}$, we measure its similarity $c_{t,m}$ with each $\boldsymbol{m}_m$ in $\mathcal{M}$:

$$c_{t,m} = \frac{\exp\left(-\|\boldsymbol{h}_t - \boldsymbol{m}_m\|^2\right)}{\sum_{k=1}^{M} \exp\left(-\|\boldsymbol{h}_t - \boldsymbol{m}_k\|^2\right)}, \quad m = 1, \dots, M.$$

Each $\boldsymbol{h}_t$ is then represented by an attention vector $\boldsymbol{c}_t = [c_{t,1}, c_{t,2}, \dots, c_{t,M}]^T$ over the memory units, and the whole $\boldsymbol{H}$ is transformed as

$$\boldsymbol{C} = \mathcal{M}(\boldsymbol{H}) = [\boldsymbol{c}_1, \dots, \boldsymbol{c}_{T'}] \in [0, 1]^{M \times T'}. \tag{1}$$

Given $\boldsymbol{C}$, one can reconstruct $\boldsymbol{H}$ as $\hat{\boldsymbol{H}} = \boldsymbol{M}\boldsymbol{C}$, and subsequently the time series segment $\boldsymbol{X}$ as

$$\hat{\boldsymbol{X}} = \mathcal{D}(\boldsymbol{M}\boldsymbol{C}). \tag{2}$$

With a set $\mathcal{X}$ of N length-T C-variate segments, the reconstruction loss is defined as $\mathcal{L}_{rec} = \frac{1}{NTC} \sum_{\boldsymbol{X} \in \mathcal{X}} \|\hat{\boldsymbol{X}} - \boldsymbol{X}\|_F^2$.

As in VQ-VAE (Van D. et al., 2017), the encoder $\mathcal{E}$, decoder $\mathcal{D}$ and memory bank $\mathcal{M}$ can be trained together end-to-end by using adversarial discriminator training, and a combination of the reconstruction loss and commitment/codebook loss: $\mathcal{L}_{\mathcal{M}} = \frac{1}{NT'd} \sum_{\boldsymbol{h}_t \in \boldsymbol{H}} \sum_{\boldsymbol{X} \in \mathcal{X}} \left[\|\mathrm{sg}\,[\boldsymbol{h}_t] - \boldsymbol{z}_t\|_2^2 + \|\boldsymbol{h}_t - \mathrm{sg}\,[\boldsymbol{z}_t]\|_2^2 \right]$, where $\mathrm{sg}[\cdot]$ is the stop-gradient operator and $\boldsymbol{z}_t = \arg\min_{\boldsymbol{m} \in \boldsymbol{M}} \|\boldsymbol{h}_t - \boldsymbol{m}\|$. The discriminator $\mathscr{D} = \mathrm{FC} \circ \mathcal{E}'$ consists of a network

$\mathcal{E}'$ (which has the same structure as the encoder) and a fully-connected (FC) layer with sigmoid activation to re-scale the output to $[0, 1]$. We use the hinge loss for the discriminator:

$$\mathcal{L}_{\mathscr{D}} = \frac{1}{NT'} \sum_{\boldsymbol{X} \in \mathcal{X}} \sum_{i=1}^{T'} [\max\{0, 1 - d_i\} + \max\{0, 1 + \hat{d}_i\}], \tag{3}$$

where $\boldsymbol{d} = [d] = \mathscr{D}(\boldsymbol{X}) \in [0, 1]^{T'}$ and $\hat{\boldsymbol{d}} = [\hat{d}] = \mathscr{D} \circ \mathcal{D}(\boldsymbol{MC}) \in [0, 1]^{T'}$ are the discriminator outputs for the original $\boldsymbol{X}$ and reconstructed $\hat{\boldsymbol{X}}$, respectively. Putting the various losses together, the stage I objective is: $\min_{\mathcal{D},\mathcal{E},\mathcal{M}} \max_{\mathscr{D}} \mathcal{L}_{rec} + \alpha\mathcal{L}_{\mathcal{M}} + \lambda\mathcal{L}_{\mathscr{D}}$, where α and λ are hyperparameters. As in GAN (Goodfellow et al., 2014), this is optimized by alternating (i) learning of $\mathscr{D}$ on $\mathcal{L}_{\mathscr{D}}$, and (ii) learning of $\{\mathcal{E}, \mathcal{D}, \mathcal{M}\}$ on

$$\mathcal{L} = \mathcal{L}_{rec} + \alpha\mathcal{L}_{M} - \frac{\lambda}{NT'} \sum_{\boldsymbol{X} \in \mathcal{X}} \sum_{i=1}^{T'} \hat{d}_i. \tag{4}$$

The whole learning procedure for Stage I is shown in Algorithm 1.

Algorithm 1 Training of Stage I.

Input: set of time series segments $\mathcal{X}$.
Output: encoder $\mathcal{E}$, decoder $\mathcal{D}$ and memory bank $\mathcal{M}$.

for $e = 1, \ldots, E$ **do**
  draw $\boldsymbol{X} \sim \mathcal{X}$;
  feed $\boldsymbol{X}$ to encoder and output $\boldsymbol{H}$;
  compute attention sequence $\boldsymbol{C}$ from (1);
  if $e \equiv 0 \mod 2$ **then**
    freeze $\mathscr{D}$, and learn $\{\mathcal{E}, \mathcal{D}, \mathcal{M}\}$ by minimizing (4);
  else
    freeze $\{\mathcal{E}, \mathcal{D}, \mathcal{M}\}$, and learn $\mathscr{D}$ by minimizing (3);
  end if
end for
return trained $\mathcal{E}$, $\mathcal{D}$, and $\mathcal{M}$.

3.2 Stage II

In stage II, the encoder $\mathcal{E}$, decoder $\mathcal{D}$, and memory bank $\mathcal{M}$ are frozen. Recall that stage I compresses the length-T time series segment $\boldsymbol{X}$ to a length-T' attention sequence $\boldsymbol{C}$. When H future values are to be forecasted, the resultant length-$(T + H)$ time series segment corresponds to an attention sequence $\hat{\boldsymbol{C}}$ of length $\lceil T'(T + H)/T \rceil$. In other words, $H' = \lceil T'(T + H)/T \rceil - T' = \lceil T'H/T \rceil$ extra attention vectors need to be predicted. Stage II uses a sequence-to-sequence predictor $\mathcal{P}$ to produce this $\hat{\boldsymbol{C}} = [\hat{\boldsymbol{c}}_1, \ldots \hat{\boldsymbol{c}}_{T'+H'}]$. The predictor can be any sequence-to-sequence model. In the experiments, we will use an LSTM (Hochreiter & Schmidhuber, 1997). $\hat{\boldsymbol{C}}$ is then obtained as

$$\hat{\boldsymbol{C}} = \text{Sofmax} \circ \mathcal{P}(\boldsymbol{C}), \tag{5}$$

where Sofmax runs over the first dimension of $\mathcal{P}(\boldsymbol{C})$ to ensure that the components of each attention vector sum to 1 over the memory units.

To train the LSTM predictor, the length-T input time series segment, together with its length-H ground-truth forecast, are concatenated to form $\boldsymbol{X}_{[:,1:T+H]}$. This is then fed into stage I to obtain the length-$(T' + H')$ ground-truth attention sequence:

$$\boldsymbol{C}^{\text{gt}} = [\boldsymbol{c}_1^{\text{gt}}, \ldots, \boldsymbol{c}_{T'+H'}^{\text{gt}}] = \mathcal{M} \circ \mathcal{E}(\boldsymbol{X}_{[:,1:T+H]}). \tag{6}$$

The LSTM is trained by minimizing the classification loss:

$$\mathcal{L}_{\text{pred}} = \frac{1}{T' + H'} \sum_{t=1}^{T'+H'} \text{BCE}\left(\hat{\boldsymbol{c}}_t, \boldsymbol{c}_t^{\text{gt}}\right), \tag{7}$$

where $\text{BCE}(\boldsymbol{u}, \boldsymbol{v})$ is the binary cross-entropy loss. The whole learning procedure for Stage II is shown in Algorithm 2.

3.3 Inference

On inference, given a new test time series segment $\boldsymbol{X}$, we first obtain its attention sequence $\boldsymbol{C}$ by using encoder $\mathcal{E}$ and memory bank $\mathcal{M}$. This is then fed into predictor $\mathcal{P}$ to get the predicted memory attention $\hat{\boldsymbol{C}}$, which is subsequently decoded by the decoder $\mathcal{D}$ to obtain the prediction $\hat{\boldsymbol{X}}_{[:,1:T+H]}$ by (2). Finally, the forecast is extracted as $\hat{\boldsymbol{X}}_{[:,T+1:T+H]}$. The procedure is shown in Algorithm 3.

Algorithm 2 Training of Stage II.

Input: set of time series segments $\mathcal{X}$, optimized encoder $\mathcal{E}$, decoder $\mathcal{D}$, and memory bank $\mathcal{M}$.
Output: predictor $\mathcal{P}$.

for $e = 1, \dots, E$ **do**
  draw $\boldsymbol{X} \in \mathcal{X}$ and obtain ground-truth forecast $\tilde{\boldsymbol{X}}$;
  compute $\boldsymbol{X}$'s attention sequence $\boldsymbol{C}$ from (1);
  compute predicted attention sequence $\hat{\boldsymbol{C}}$ from (5);
  compute ground-truth attention $\boldsymbol{C}^{\text{gt}}$ of $[\boldsymbol{X}; \tilde{\boldsymbol{X}}]$ from (6);
  learn $\mathcal{P}$ by minimizing (7);
end for
return trained $\mathcal{M}$,

Algorithm 3 Inference.

Input: new time series segment $\boldsymbol{X}$.
Output: forecast result $\hat{\boldsymbol{X}}_{[:,T+1,T+H]}$.

compute attention sequence $\boldsymbol{C}$ of $\boldsymbol{X}$ from (1);
compute $\hat{\boldsymbol{C}}$ from $\mathcal{P}$ using (5);
compute prediction $\hat{\boldsymbol{X}}$ by (2);
return $\hat{\boldsymbol{X}}_{[:,T+1:T+H]}$.

3.4 Discussion

MATS is inspired by VQ-VAE and VQGAN (Esser et al., 2021). This allows stage I to focus only on the extraction of local temporal patterns, while stage II on capturing long-term dependencies. However, in VQ-VAE and VQGAN, stage I outputs the discrete index of the most similar token in the codebook. Hence, only one codebook entry is used in representing the input. Moreover, an embedding of codebook indices needs to be learned before feeding into the transformer in stage II. On the other hand, the proposed memory bank outputs a sequence of attentions over the memory units. As demonstrated in the success of transformers, attention is more flexible and allows a weighted combination of codebooks to be used. This encourages each codebook (or memory unit) to learn more local patterns and empirically leads to better performance. Moreover, as $\boldsymbol{C}$ is continuous-valued, it can be directly fed into the predictor without learning an extra embedding. Empirically comparison will be shown in Section 4.1. Besides, compared to LTSF methods that do not use memory or codebook (such as DLinear (Zeng et al., 2022) and time series transformers), the use of memory allows extraction of informative patterns across windows in all training samples, instead of just from a given window from the current sample.

Note that stage I compresses the length-T input time series segment to a length-T' attention sequence. Similarly, in stage II, instead of directly predicting a length-H time series segment, it predicts a length-H' attention sequence. Typically, $T' \ll T$ and $H' \ll H$. Thus, for the stage II predictor, the input and output sequences are shorter than the raw time series input and prediction horizon, respectively, making the prediction task easier. This will also be empirically verified in Section 4.2.1.

4 Experiments

In this section, experiments are performed on the following commonly-used multivariate datasets[1] (Zhou et al., 2021; Wu et al., 2021; Zhou et al., 2022; Zeng et al., 2022): (i) Electricity, which

[1] Electricity is downloaded from `https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014`, Exchange from `https://github.com/laiguokun/multivariate-time-series-data`, Traffic from `http://pems.dot.ca.gov`, Weather from `https://www.bgc-jena.mpg.de/wetter/`, ETT from `https://github.com/zhouhaoyi/ETDataset`, ILI from `https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html`.

includes electricity consumption of 321 clients. Following Zhou et al. (2021), the data is converted to hourly consumption over two years; (ii) Exchange, which describes the daily exchange rates of Australia, British, Canada, Switzerland, China, Japan, New Zealand, and Singapore; (iii) Traffic, which records the hourly road occupancy rates generated by sensors on San Francisco Bay area freeways; (iv) Weather, which records 21 meteorological indicators at 10-minute intervals in 2020; (v) ETT, which contains two years of electricity transformer temperature data collected in China (Zhou et al., 2021). It has four subsets: ETTh1, ETTh2 are collected from two counties at 1-hour intervals, while ETTm1, ETTm2 are collected at 15-minute intervals; (vi) ILI, which contains weekly records of the ratio of influenza-like illness (ILI) from 2002-2021 in the United States. Datasets are summarized in Appendix A.1. As in Zhou et al. (2021), we also perform experiments on univariate variants of these datasets, which are formed by extracting the last variate from the multivariate datasets. Following Wu et al. (2021); Zhou et al. (2022), data are normalized to have zero mean and unit variance. Each dataset is split in chronological order into the training, validation, and test sets of size 7:1:2, except that for ILI follows 6:2:2.

In MATS, stage I uses a three-layer CNN as the encoder, decoder, and discriminator. The memory bank has a dimensionality of 64 and a size of 16. We use the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 10^{-4}. We use a two-layer LSTM with hidden dimension 1024, dropout probability 0.5, and the AdamW optimizer (Loshchilov & Hutter, 2018) for the predictor in stage II, The batch size in both stages is 64. The detailed architecture and hyperparameter setup are in Appendix A.2. To demonstrate the advantage of using memory attention, we introduce a variant called VQ-LSTM, which replaces the memory bank in MATS with a VQ codebook from VQ-VAE and adds a learnable codebook index embedding as in VQGAN.

MATS and its variant VQ-LSTM are compared with the following families of baselines: (i) time series transformers, including FEDformer (Zhou et al., 2022), Autoformer (Wu et al., 2021), Informer (Zhou et al., 2021), Pyraformer (Liu et al., 2021b), LogTrans (Li et al., 2019), and Reformer (Kitaev et al., 2020).; (ii) recurrent neural networks (RNN), including LSTM (Hochreiter & Schmidhuber, 1997) and LSTMa (Bahdanau et al., 2015); (iii) convolutional neural networks (CNN), including TCN (Bai et al., 2018) and LSTNet (Lai et al., 2018). and (iv) recent SOTA methods DLinear (Zeng et al., 2022) and DeepTIMe (Woo et al., 2022); For the univariate time series, we also include (v) the standard statistical model of ARIMA (Anderson, 1976); and (vi) N-Beats (Oreshkin et al., 2019). Finally, following Zhou et al. (2022), we add a baseline called (vii) Closest Repeat (Repeat-C), which naively uses the last value in the window as the prediction. All algorithms are in PyTorch and run on an NVIDIA Tesla V100 32G GPU.

For stage I, we set the input window size $T = 192$ (except for ILI, which is set to 60). Following Wu et al. (2021); Zhou et al. (2022), in stage II, we set $T = 96$ (for ILI, $T = 36$), and the prediction horizon $H \in \{96, 192, 336, 720\}$ (for ILI, $H \in \{24, 36, 48, 60\}$). For performance evaluation, we use (i) mean squared error MSE $= \frac{1}{NHC} \sum_{\boldsymbol{X} \in \mathcal{X}} \sum_{i=1}^{H} \|\hat{\boldsymbol{x}}_{T+i} - \tilde{\boldsymbol{x}}_{T+i}\|_2^2$, where N is the number of segments in the test set, and (ii) mean absolute error MAE $= \frac{1}{NHC} \sum_{\boldsymbol{X} \in \mathcal{X}} \sum_{i=1}^{H} \|\hat{\boldsymbol{x}}_{T+i} - \tilde{\boldsymbol{x}}_{T+i}\|_1$.

4.1 Results

The MSE results on the multivariate and univariate time series are shown in Tables 1 and 2, respectively. For baselines DLinear, DeepTIMe, LSTNet, LSTMa, TCN, and NBeats, we use the implementations from the corresponding authors; for ARIMA, we use the implementation from the Python package statsmodels; for baselines FEDFormer, Autoformer, Informer, Pyraformer, LogTrans, Reformer, LSTM, DLinear, and Repeat-C, we copy their results from the corresponding papers Wu et al. (2021); Zhou et al. (2022); Zeng et al. (2022). As expected, the error generally increases with the prediction horizon across all methods. MATS outperforms all the baselines most of the time. In particular, MATS beats LSTNet, which also uses CNN and RNN but is trained end-to-end. Moreover, the variant VQ-LSTM is often the second-best algorithm. These show the superiority of the proposed two-stage approach that helps the model learn useful local patterns by an adversarial reconstruction task in stage I without concerning long-term dependencies. Moreover, MATS outperforms VQ-LSTM almost all the time, demonstrating the efficiency of using memory attention over codebook indices.

Figure 2 shows the forecasting results on the first test sample from the univariate Electricity (with $T = 96$ and $H = 96$). More samples can be seen in Appendix B.1. To avoid clutterness, we only

Table 1: MSE forecasting results on multivariate time series. Results of baselines marked with superscript ◇ are copied from Wu et al. (2021); Zhou et al. (2022); Zeng et al. (2022). The best results are in bold, and the second best are underlined.

	H	Electricity	Exchange	Traffic	Weather	ETTh1	ETTh2	ETTm1	ETTm2		ILI
MATS	96	**0.156**	**0.034**	**0.516**	**0.105**	**0.301**	**0.222**	**0.185**	**0.109**	24	**1.143**
	192	**0.165**	**0.049**	**0.549**	**0.133**	**0.351**	**0.239**	**0.306**	**0.130**	36	**1.647**
	336	**0.168**	**0.073**	**0.565**	**0.156**	**0.375**	**0.283**	**0.348**	**0.162**	48	2.343
	720	**0.179**	**0.135**	**0.602**	**0.196**	**0.407**	**0.315**	**0.368**	**0.222**	60	**2.228**
VQ-LSTM	96	0.187	0.045	0.576	0.119	0.422	0.226	0.335	0.113	24	1.849
	192	0.187	0.062	0.596	0.163	0.461	0.270	0.402	0.147	36	1.838
	336	0.189	0.110	0.610	0.176	0.491	0.338	0.410	0.189	48	**2.161**
	720	0.200	0.363	0.636	0.247	0.514	0.459	0.480	0.278	60	3.057
FEDFormer◇	96	0.193	0.148	0.587	0.217	0.376	0.346	0.379	0.203	24	3.228
	192	0.201	0.271	0.604	0.276	0.420	0.429	0.426	0.269	36	2.679
	336	0.214	0.460	0.621	0.339	0.459	0.496	0.445	0.325	48	2.622
	720	0.246	1.195	0.626	0.403	0.506	0.463	0.543	0.421	60	2.857
Autoformer◇	96	0.201	0.197	0.613	0.266	0.449	0.358	0.505	0.255	24	3.483
	192	0.222	0.300	0.616	0.307	0.500	0.456	0.553	0.281	36	3.103
	336	0.231	0.509	0.622	0.359	0.521	0.482	0.621	0.339	48	2.669
	720	0.254	1.447	0.660	0.419	0.514	0.515	0.671	0.433	60	2.770
Informer◇	96	0.274	0.847	0.719	0.300	0.865	3.755	0.672	0.365	24	5.764
	192	0.296	1.204	0.696	0.598	1.008	5.602	0.795	0.533	36	4.755
	336	0.300	1.672	0.777	0.578	1.107	4.721	1.212	1.363	48	4.763
	720	0.373	2.478	0.864	1.059	1.181	3.647	1.166	3.379	60	5.264
Pyraformer◇	96	0.386	1.748	0.867	0.622	0.664	0.645	0.543	0.435	24	7.394
	192	0.378	1.874	0.869	0.739	0.790	0.788	0.557	0.730	36	7.551
	336	0.376	1.943	0.881	1.004	0.891	0.907	0.754	1.201	48	7.662
	720	0.376	2.085	0.896	1.420	0.963	0.963	0.908	3.625	60	7.931
LogTrans◇	96	0.258	0.968	0.684	0.458	0.878	2.116	0.600	0.768	24	4.480
	192	0.266	1.040	0.685	0.658	1.037	4.315	0.837	0.989	36	4.799
	336	0.280	1.659	0.734	0.797	1.238	1.124	1.124	1.334	48	4.800
	720	0.283	1.941	0.717	0.869	1.135	3.188	1.153	3.048	60	5.278
Reformer◇	96	0.312	1.065	0.732	0.689	0.837	2.626	0.538	0.658	24	4.400
	192	0.348	1.188	0.733	0.752	0.923	11.12	0.658	1.078	36	4.783
	336	0.350	1.357	0.742	0.639	1.097	9.323	0.898	1.549	48	4.832
	720	0.340	1.510	0.755	1.130	1.257	3.874	1.102	2.631	60	4.882
LSTM◇	96	0.375	1.453	0.843	0.369	0.702	1.671	1.392	2.041	24	5.914
	192	0.442	1.846	0.847	0.416	1.212	4.117	1.339	2.249	36	6.631
	336	0.429	2.136	0.853	0.455	1.424	3.434	1.740	2.568	48	6.736
	720	0.980	2.984	1.500	0.535	1.960	3.963	2.736	2.720	60	6.870
LSTMa	96	0.377	0.890	2.078	0.330	0.815	2.079	1.200	0.747	24	4.312
	192	0.404	1.358	1.985	0.494	1.029	1.761	1.148	2.041	36	4.300
	336	0.762	1.781	1.786	0.605	1.115	2.596	1.148	0.969	48	4.305
	720	1.309	2.326	1.748	0.594	1.299	2.932	1.119	2.541	60	4.418
TCN	96	0.985	3.004	1.438	0.615	1.038	3.307	0.957	3.041	24	6.624
	192	0.996	3.048	1.463	0.629	1.070	3.359	0.969	3.072	36	6.858
	336	1.000	3.113	1.479	0.639	1.085	3.443	0.990	3.105	48	6.968
	720	1.438	3.150	1.499	0.639	1.089	3.626	1.018	3.135	60	7.127
LSTNet	96	0.680	1.551	1.107	0.594	1.465	3.567	1.999	3.142	24	6.026
	192	0.725	1.477	1.157	0.560	1.997	3.242	2.762	3.154	36	5.340
	336	0.828	1.507	1.216	0.597	2.665	2.544	1.257	3.160	48	6.080
	720	0.957	2.285	1.481	0.618	2.143	4.625	1.917	3.171	60	5.548
DLinear◇	96	0.194	0.078	0.650	0.196	0.386	0.295	0.345	0.183	24	2.398
	192	0.193	0.159	0.598	0.237	0.437	0.452	0.380	0.260	36	2.646
	336	0.206	0.274	0.605	0.283	0.481	0.504	0.413	0.336	48	2.614
	720	0.242	0.558	0.645	0.345	0.519	0.577	0.474	0.423	60	2.804
DeepTIMe	96	0.198	0.077	0.661	0.197	0.395	0.325	0.347	0.188	24	2.361
	192	0.196	0.151	0.605	0.240	0.445	0.427	0.384	0.265	36	2.328
	336	0.209	0.248	0.611	0.285	0.487	0.492	0.416	0.325	48	2.675
	720	0.245	0.689	0.656	0.352	0.524	0.716	0.482	0.486	60	2.904
Repeat-C◇	96	1.588	0.081	2.723	0.259	1.295	0.432	1.214	0.266	24	6.587
	192	1.595	0.167	2.756	0.309	1.325	0.534	1.261	0.340	36	7.130
	336	1.617	0.305	2.791	0.377	1.323	0.591	1.283	0.412	48	6.575
	720	1.647	0.823	2.811	0.465	1.339	0.588	1.319	0.521	60	5.893

compare with the forecasts of (i) FedFormer, the SOTA time series transformer; and (ii) two recent SOTAs: DLinear and DeepTIMe; and (iii) N-Beats, a popular univariate LTSF method. As can be seen, MATS produces better forecast than others.[2]

[2]For this sample, the MSEs obtained are as follows. FEDFormer: 0.263; DLinear: 0.123; DeepTIMe: 0.142; NBeats: 0.185; and MATS: 0.097.

Table 2: MSE forecasting results on univariate time series. Results of baselines marked with superscript ⋄ are copied from Wu et al. (2021); Zhou et al. (2022); Zeng et al. (2022). The best results are in bold, and the second best are underlined.

	H	Electricity	Exchange	Traffic	Weather	ETTh1	ETTh2	ETTm1	ETTm2		ILI
MATS	96	**0.177**	**0.019**	**0.135**	**0.000**	**0.024**	0.098	**0.009**	**0.018**	24	**0.629**
	192	**0.252**	**0.044**	**0.149**	**0.001**	**0.033**	**0.115**	**0.015**	**0.038**	36	0.681
	336	**0.260**	**0.069**	**0.155**	**0.001**	**0.044**	**0.149**	**0.020**	**0.058**	48	0.760
	720	**0.286**	**0.262**	**0.184**	**0.001**	**0.055**	**0.191**	**0.031**	**0.078**	60	**0.740**
VQLSTM	96	0.362	0.020	0.249	0.001	0.029	**0.087**	0.012	0.030	24	0.786
	192	0.397	0.059	0.244	0.001	0.042	0.118	0.018	0.049	36	0.929
	336	0.433	0.124	0.234	0.001	0.048	0.171	0.026	0.071	48	1.142
	720	0.465	0.359	0.247	0.001	0.064	0.192	0.043	0.101	60	0.969
FEDFormer⋄,	96	0.253	0.154	0.207	0.006	0.079	0.128	0.033	0.067	24	0.708
	192	0.282	0.286	0.205	0.006	0.104	0.185	0.058	0.102	36	**0.584**
	336	0.346	0.511	0.219	0.004	0.119	0.231	0.084	0.130	48	**0.717**
	720	0.422	1.301	0.244	0.006	0.142	0.278	0.102	0.178	60	0.855
Autoformer⋄	96	0.341	0.241	0.246	0.011	0.071	0.153	0.056	0.065	24	0.948
	192	0.345	0.300	0.266	0.008	0.114	0.204	0.081	0.118	36	0.634
	336	0.406	0.509	0.263	0.006	0.107	0.246	0.076	0.154	48	0.791
	720	0.565	1.260	0.269	0.009	0.126	0.268	0.110	0.182	60	0.874
Informer⋄	96	0.258	1.327	0.257	0.004	0.193	0.213	0.109	0.088	24	5.282
	192	0.285	1.258	0.299	0.002	0.217	0.227	0.151	0.132	36	4.554
	336	0.336	2.179	0.312	0.004	0.202	0.242	0.427	0.180	48	4.273
	720	0.607	1.280	0.366	0.003	0.183	0.291	0.438	0.300	60	5.214
Pyraformer⋄	96	0.274	0.230	0.166	0.007	0.245	0.149	0.079	0.078	24	5.082
	192	0.279	0.648	0.168	0.006	0.272	0.171	0.180	0.113	36	4.141
	336	0.321	2.304	0.174	0.006	0.288	0.249	0.189	0.176	48	4.554
	720	0.532	2.378	0.221	0.007	0.307	0.219	0.378	0.232	60	5.003
LogTrans⋄	96	0.288	0.237	0.226	0.317	0.283	0.217	0.049	0.075	24	3.607
	192	0.432	0.738	0.314	0.408	0.234	0.281	0.157	0.129	36	2.407
	336	0.430	2.018	0.387	0.453	0.386	0.293	0.289	0.154	48	3.106
	720	0.491	2.405	0.437	0.491	0.475	0.218	0.430	0.160	60	3.698
Reformer⋄	96	0.275	0.298	0.313	0.012	0.532	1.411	0.296	0.076	24	3.838
	192	0.304	0.777	0.386	0.010	0.568	5.658	0.429	0.132	36	2.934
	336	0.370	1.833	0.423	0.013	0.635	4.777	0.585	0.160	48	3.755
	720	0.460	1.203	0.378	0.011	0.762	2.042	0.782	0.168	60	4.162
LSTMa	96	0.953	1.327	3.909	0.006	0.589	0.607	0.618	0.786	24	2.056
	192	0.927	2.628	3.529	0.013	1.023	1.035	1.033	1.053	36	2.757
	336	1.212	2.785	3.146	0.010	1.328	1.347	1.482	1.166	48	3.104
	720	1.511	3.094	3.152	0.013	1.697	1.463	1.701	1.406	60	3.124
DLinear⋄	96	0.372	0.099	0.305	0.005	0.062	0.129	0.034	0.372	24	1.144
	192	0.352	0.203	0.256	0.005	0.080	0.183	0.065	0.352	36	1.161
	336	0.380	0.343	0.251	0.006	0.104	0.232	0.078	0.380	48	1.304
	720	0.422	0.827	0.297	0.007	0.175	0.318	0.104	0.422	60	1.888
DeepTIMe	96	0.382	0.086	0.298	0.004	0.060	0.131	0.034	0.073	24	0.931
	192	0.363	0.175	0.246	0.001	0.084	0.187	0.051	0.106	36	0.973
	336	0.389	0.295	0.237	0.002	0.109	0.236	0.073	0.139	48	1.105
	720	0.438	0.658	0.274	0.002	0.277	0.347	0.114	0.191	60	1.221
ARIMA	96	1.343	0.112	1.140	0.003	0.093	0.355	0.083	2.238	24	2.107
	192	1.523	0.304	1.389	0.011	0.207	0.830	0.272	7.871	36	2.815
	336	2.108	0.736	1.577	0.036	0.629	2.273	0.994	23.62	48	4.043
	720	4.699	1.871	1.748	0.316	1.142	8.880	9.496	16.09	60	5.520
N-BEATS	96	0.350	0.156	0.162	0.003	0.108	0.135	0.043	0.070	24	1.893
	192	0.380	0.669	0.165	0.003	0.248	0.166	0.214	0.120	36	1.231
	336	0.401	0.611	0.166	0.004	0.134	0.245	0.273	0.165	48	1.395
	720	0.489	1.111	0.204	0.005	0.243	0.319	0.153	0.207	60	1.975
Repeat-C⋄	96	1.639	0.088	3.628	0.001	0.069	0.296	0.033	0.228	24	1.487
	192	1.650	0.189	3.607	0.002	0.092	0.337	0.049	0.257	36	1.325
	336	1.746	0.372	3.596	0.002	0.114	0.390	0.065	0.287	48	1.283
	720	1.824	1.009	3.551	0.002	0.129	0.437	0.089	0.332	60	1.339

4.2 Ablation Studies

4.2.1 Memory Attention Sequence Length T'

In stage I, the encoder encodes a length-T raw time series to a length-T' memory attention sequence. Table 3 shows the MSE on the Electricity and Exchange with varying T' ($T = 96, H = 96$). As can be seen, for univariate datasets, $T' = 4$ shows the best result, while for multivariate datasets, $T = 32$ shows the best result. When T' is large, stage I needs to represent longer input sequences and stage II needs to predict longer output sequences, making both harder to learn and the performance degrades. On the other hand, when T' is too small, the attention sequence may not be representative

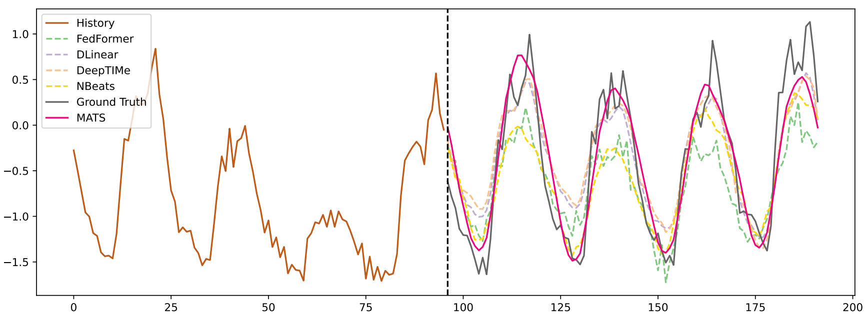


Figure 2: Example forecasting result on univariate Electricity.

enough to represent the input time series segment (as in Multivariate Electricity). As a compromise, we set $T/T' = 12$ for all the datasets.

Table 3: MSE with T' on the Electricity and Exchange (with $T = 96, H = 96$).

T'	Univariate Electricity	Univariate Exchange	Multivariate Electricity	Multivariate Exchange
2	0.176	0.022	0.185	0.036
4	**0.170**	**0.017**	0.171	0.036
8	0.177	0.019	0.156	0.034
16	0.194	0.019	0.155	0.035
24	0.217	0.019	0.153	0.034
32	0.212	0.021	**0.135**	**0.033**
48	0.218	0.022	0.136	0.035
96	0.230	0.023	0.140	0.035

Table 4: LSTM versus Transformer MSE on Electricity and Exchange.

	H	Univariate Electricity	Univariate Exchange	Multivariate Electricity	Multivariate Exchange
LSTM	96	**0.177**	**0.019**	**0.156**	**0.034**
	192	**0.252**	**0.044**	**0.165**	**0.049**
	336	**0.260**	**0.069**	**0.168**	**0.073**
	720	**0.286**	**0.262**	**0.179**	**0.135**
Transformer	96	0.809	0.048	0.796	0.068
	192	0.827	0.080	0.804	0.090
	336	0.852	0.109	0.809	0.131
	720	0.864	0.386	0.821	0.256

4.2.2 Size of Memory Bank

Figure 3 shows the MSE with varying memory size m on the univariate Electricity. As can be seen, when the memory size is too small (equal to 2), the memory bank cannot store all local patterns, and so the performance is worse. On the other hand, when the memory is sufficiently large, further increasing the memory size brings no additional improvement to performance.

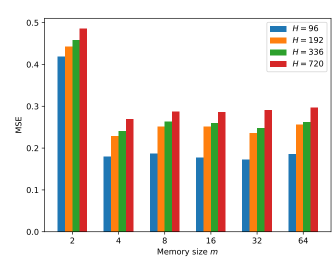


Figure 3: MSE with memory size m on the univariate Electricity.

4.2.3 Transformer versus LSTM

Table 4 compares the use of LSTM versus transformer as the stage II predictor on the univariate Electricity and Exchange datasets (with $T = 96$). As can be seen, LSTM outperforms the transformer. It may be because transformers are more appropriate when the input sequence is long, while here, we have compressed the input time series to shorter attention sequences of length $T' + H'$.

5 Conclusion

In this paper, we proposed a novel long-range time series forecasting model called **M**emory **A**ttention for **T**ime-**S**eries forecasting (MATS). It uses a memory bank to extract abundant short-term temporal patterns, and represents an input time series as a short sequence of memory attentions. The use of attention allows a flexible representation, and its shorter sequence length enables the model to learn long-term dependencies more easily. Extensive experiments demonstrate that MATS outperforms a variety of recent SOTA methods almost all the time.

REFERENCES

O. Anderson. Time-Series. 2nd edn., 1976.

D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*, 2015.

S. Bai, J. Kolter, and V. Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. Preprint arXiv:1803.01271, 2018.

A. Borovykh, S. Bohte, and C. Oosterlee. Conditional Time Series Forecasting with Convolutional Neural Networks. *Stat*, 2017.

G. Box and G. Jenkins. Some Recent Advances in Forecasting and Control. *Journal of the Royal Statistical Society*, 1968.

G. Box and D. Pierce. Distribution of Residual Autocorrelations in Autoregressive-integrated Moving Average Time Series Models. *Journal of the American statistical Association*, 1970.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language Models are Few-shot Learners. In *Advances in Neural Information Processing Systems*, 2020.

C. Deb, F. Zhang, J. Yang, S. Lee, and K. Shah. A Review on Time Series Forecasting Techniques for Building Energy Consumption. *Renewable and Sustainable Energy Reviews*, 2017.

S. Elsayed, D. Thyssens, A. Rashed, H. Jomaa, and L. Schmidt-Thieme. Do We Really Need Deep Learning Models for Time Series Forecasting? Preprint arXiv:2101.02118, 2021.

P. Esser, R. Rombach, and B. Ommer. Taming Transformers for High-Resolution Image Synthesis. In *Computer Vision and Pattern Recognition*, 2021.

J. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, 2001.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *Advances in Neural information processing systems*, 2014.

S. Hochreiter and J. Schmidhuber. Long Short-term Memory. *Neural computation*, 1997.

J. Kenton and L. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, 2019.

D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

N. Kitaev, L. Kaiser, and A. Levskaya. Reformer: The Efficient Transformer. In *International Conference on Learning Representations*, 2020.

G. Lai, W. Chang, Y. Yang, and H. Liu. Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks. In *International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018.

S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y. Wang, and X. Yan. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. In *Advances in Neural Information Processing Systems*, 2019.

M. Liu, A. Zeng, Z. Xu, Q. Lai, and Q. Xu. Time Series Is A Special Sequence: Forecasting With Sample Convolution and Interaction. Preprint arXiv:2106.09305, 2021a.

S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. Liu, and S. Dustdar. Pyraformer: Low-Complexity Pyramidal Attention for Long-Range Time Series Modeling and Forecasting. In *International Conference on Learning Representations*, 2021b.

I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2018.

D. Maddix, Y. Wang, and A. Smola. Deep Factors with Gaussian Processes for Forecasting. Preprint arXiv:1812.00098, 2018.

Y. Matsubara, Y. Sakurai, W. Van Panhuis, and C. Faloutsos. FUNNEL: Automatic Mining of Spatially Coevolving Epidemics. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014.

B. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2019.

S. Papadimitriou and P. Yu. Optimal Multi-scale Patterns in Time Series Streams. In *ACM SIGMOD International Conference on Management of Data*, 2006.

Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. Cottrell. A Dual-stage Attention-based Recurrent Neural Network for Time Series Prediction. In *International Joint Conference on Artificial Intelligence*, 2017.

S. Rangapuram, M. W Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. Deep State Space Models for Time Series Forecasting. In *Advances in Neural Information Processing Systems*, 2018.

D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. *International Journal of Forecasting*, 2020.

R. Sen, H. Yu, and I. Dhillon. Think Globally, Act Locally: A Deep Neural Network Approach to High-dimensional Time Series Forecasting. In *Advances in Neural Information Processing Systems*, 2019.

S. Shih, F. Sun, and H. Lee. Temporal Pattern Attention for Multivariate Time Series Forecasting. *Machine Learning*, 2019.

H. Song, D. Rajan, J. Thiagarajan, and A. Spanias. Attend and Diagnose: Clinical Time Series Analysis Using Attention Models. In *AAAI conference on Artificial Intelligence*, 2018.

Aaron Van D., O. Vinyals, et al. Neural Discrete Representation Learning. In *Advances in Neural Information Processing Systems*, 2017.

A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. In *ISCA Speech Synthesis Workshop*, 2016.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Ł. Kaiser, and I. Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, 2017.

R. Wen, K. Torkkola, B. Narayanaswamy, and D. Madeka. A Multi-horizon Quantile Recurrent Forecaster. Preprint arXiv:1711.11053, 2017.

J. Weston, S. Chopra, and A. Bordes. Memory Networks. In *International Conference on Learning Representations*, 2015.

G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi. DeepTIMe: Deep Time-Index Meta-Learning for Non-Stationary Time-Series Forecasting. Preprint arXiv:2207.06046, 2022.

H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In *Advances in Neural Information Processing Systems*, 2021.

S. Wu, X. Xiao, Q. Ding, P. Zhao, Y. Wei, and J. Huang. Adversarial Sparse Transformer for Time Series Forecasting. In *Advances in Neural Information Processing Systems*, 2020.

R. Yu, S. Zheng, A. Anandkumar, and Y. Yue. Long-term Forecasting Using Tensor-train RNNs. *Arxiv*, 2017.

M. Zeiler, D. Krishnan, G. Taylor, and R. Fergus. Deconvolutional Networks. In *Computer Society Conference on computer vision and pattern recognition*, 2010.

A. Zeng, M. Chen, L. Zhang, and Q. Xu. Are Transformers Effective for Time Series Forecasting? Preprint arXiv:2205.13504, 2022.

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 *AAAI Conference on Artificial Intelligence*, 2021.

T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In *International Conference on Machine Learning*, 2022.

Y. Zhu and D. Shasha. Statstream: Statistical Monitoring of Thousands of Data Streams in Real Time. In *International Conference on Very Large Databases*, 2002.

A APPENDIX

A.1 DATASETS SUMMATION

Table 5 summarized the datasets used in the experiments.

Table 5: Summary of the datasets.

	#variates	#timesteps
Electricity	321	26,304
Exchange	8	7,588
Traffic	862	17,544
Weather	21	52,696
ETTh1, ETTh2	7	17,420
ETTm1, ETTm2	7	69,680
ILI	7	966

A.2 DETAILED EXPERIMENT SETUP

Table 6 shows the detailed training setup of MATS. λ is calculated as in VQGAN. For all datasets, we use the same encoder, decoder and discriminator architecture. Table 7 shows the architecture of the encoder and the discriminator. Table 8 shows the architecture of the decoder. C in the two tables denotes the number of variates and differs between different datasets. The transformer variant uses a 2-layer, 8-head transformer and each head's hidden dimension is 128.

Table 6: Detailed hyperparameters.

Dataset	Univariate/ Multivariate	#Epoch of Stage I	Learning Rate of Stage I	#Epoch of Stage II	Learning Rate of Stage II	α
Electricity	Multivariate	1,000	1×10^{-4}	500	1×10^{-4}	1.0
	Univariate	1,000	1×10^{-4}	500	1×10^{-4}	1.0
Exchange	Multivariate	1,000	1×10^{-4}	500	1×10^{-4}	1.0
	Univariate	1,000	1×10^{-4}	500	1×10^{-4}	1.0
Traffic	Multivariate	1,000	1×10^{-4}	200	1×10^{-4}	1.0
	Univariate	1,000	1×10^{-4}	200	1×10^{-4}	1.0
Weather	Multivariate	1,000	1×10^{-4}	100	1×10^{-4}	1.0
	Univariate	1,000	1×10^{-4}	100	1×10^{-4}	1.0
ETTm1	Multivariate	1,000	1×10^{-4}	500	1×10^{-4}	1.0
	Univariate	1,000	1×10^{-4}	500	1×10^{-4}	1.0
ETTm2	Multivariate	1,000	1×10^{-4}	100	1×10^{-4}	1.0
	Univariate	1,000	1×10^{-4}	100	1×10^{-4}	1.0
ETTh1	Multivariate	1,000	1×10^{-4}	500	1×10^{-4}	1.0
	Univariate	1,000	1×10^{-4}	500	1×10^{-4}	1.0
ETTh2	Multivariate	1,000	1×10^{-4}	100	1×10^{-4}	1.0
	Univariate	1,000	1×10^{-4}	100	1×10^{-4}	1.0
ILI	Multivariate	1,000	1×10^{-4}	500	1×10^{-3}	1.0
	Univariate	1,000	1×10^{-4}	1,000	1×10^{-3}	1.0

Table 7: Architecture of Encode and Discriminator.

Layer	Operator	Parameters
1	Convolution	in_channel=C, out_channel=128, kernel_size=4, stride=2, padding=1
2	Tanh	-
3	Dropout	dropout_rate=0.1
4	Convolution	in_channel=128, out_channel=64, kernel_size=4, stride=2, padding=1
5	Tanh	-
6	Dropout	dropout_rate=0.1
7	Convolution	in_channel=64, out_channel=64, kernel_size=5, stride=3, padding=1
8	Tanh	-

Table 8: Architecture of Decoder.

Layer	Operator	Parameters
1	DeConvolution	in_channel=64, out_channel=64, kernel_size=5, stride=3, padding=1
2	Tanh	-
3	Dropout	dropout_rate=0.1
4	DeConvolution	in_channel=64, out_channel=128, kernel_size=4, stride=2, padding=1
5	Tanh	-
6	Dropout	dropout_rate=0.1
7	DeConvolution	in_channel=128, out_channel=C, kernel_size=4, stride=2, padding=1

A.3 Additional Experiment Results

This section further provide experiment results on nine benchmark datasets in term of MAE. Table 9 and Table 10 show the results on multivariate and univariate benchmarks respectively. Baselines without superscript $\diamond$ are from Wu et al. (2021); Zhou et al. (2022); Zeng et al. (2022).

A.4 Ablation on Discriminator

We also analyze the performance grain from the discriminator $\mathscr{D}$. Table 11 shows that discriminator $\mathscr{D}$ can improve the performance of MATS in many cases, especially in very long time prediction ($H = 720$) on complex dataset (univariate and multivariate Electricity).

A.5 Ablation on memory size for more datasets

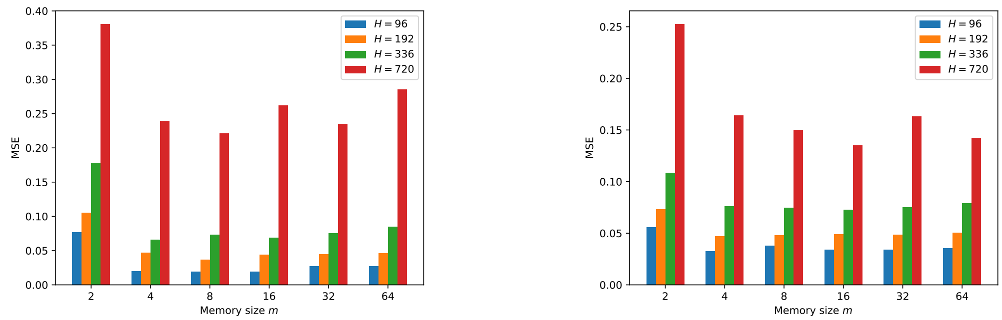
We provide additional ablation studies on memory size for univariate Exchange and multivariate Exchange. We can observe the same phenomenon as Section 4.2.2 from Figure 4.

B Visualization

B.1 Visualization on more test samples

In this section, we supply more forecasting result on univariate Electricity. We totally pick 20 samples for every 160 time steps. All visualization results are shown in Figure 5 and Figure 6. Among those examples, we can observe that i) in Figure 5(a), Figure 5(b), Figure 5(e), Figure 5(f), Figure 6(b), Figure 6(f), Figure 6(g), Figure 6(h), and Figure 6(i) MATS performs good prediction results and predicts curves very close to the ground truth curves; ii) in Figure 5(c), Figure 5(d), Figure 5(f), Figure 5(h), Figure 5(j), Figure 6(c), Figure 6(d) Figure 6(e), and Figure 6(j), MATS cannot fit very close to the baseline but performs similarly to baselines; iii) Figure 5(g), Figure 5(i) shows some fail cases that shows larger margins to ground truth curves compared to baselines.

B.2 Visualization on toy dataset



(a) MSE with memory size m on univariate Exchange.

(b) MSE with memory size m on multivariate Exchange.

Figure 4: MSE with memory size m on Exchange.

The memory bank $\mathcal{M}$ and the representation of time series as shorter sequences of memory attentions are keys to MATS. To figure out how they represent a time series, we first add in Appendix B.2 an illustration with a synthetic univariate time series: $x_t = \sin(0.08(t-1)+\pi/2)+0.6\sin(0.2(t-1)+\pi)$, where $t = 1, \dots, 10240$. To reduce redundancy of the memory units, the memory size M is set to 4.

The training set $\mathcal{X}$ has N length-96 time series segments. After Stage I training, we compute the average memory attention value as follows. Each of the N training time series $\boldsymbol{x}$ is fed through the encoder and memory bank to obtain $\{\mathcal{M} \circ \mathcal{E}(\boldsymbol{x})\}_{\boldsymbol{x}\in\mathcal{X}}$, where each $\mathcal{M} \circ \mathcal{E}(\boldsymbol{x}) \in [0,1]^{M\times T'}$ (and $T' = 8$). This is then averaged over the time dimension to obtain $\bar{\boldsymbol{c}} = [\bar{c}_1, \bar{c}_2, \dots, \bar{c}_M] \in [0,1]^M$, where $\bar{c}_i = \frac{1}{NT'}\sum_{t=1}^{T'}\sum_{\boldsymbol{x}\in\mathcal{X}}[\mathcal{M} \circ \mathcal{E}(\boldsymbol{x})]_{t,i}$.

We then compute the average memory attention for each memory unit i as $\boldsymbol{C}^i = [\boldsymbol{c}_1^i, \dots, \boldsymbol{c}_{T'}^i] \in [0,1]^{M\times T'}$, where $c_{t,i}^i = \bar{c}_i$ and $c_{t,j}^i = 0$ for $j \neq i$. This is then fed to decoder $\mathcal{D}$ (using Equation 2). The decoder outputs are shown in Figure 7(a). As can be seen, they all represent different local patterns.

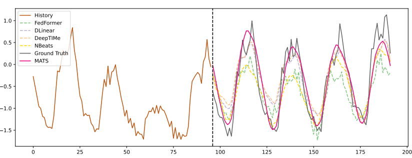
To further study how the memory units jointly represent the time series, we feed in the first training sample and obtain the memory attention sequence $\boldsymbol{C} = [\boldsymbol{c}_1, \dots, \boldsymbol{c}_{T'}] \in [0,1]^{M\times T'}$ using Equation 1. For each column vector $\boldsymbol{c}_t \in [0,1]^M$ of $\boldsymbol{C}$ (i.e. the memory attention), we keep only its top-k values, and set the remaining entries to zero. Let the resultant memory attention matrix be $\boldsymbol{C}^{(k)} = [\boldsymbol{c}_1^{(k)}, \dots, \boldsymbol{c}_{T'}^{(k)}]$. Setting $k = 1, 2, 3, 4$, we then feed them into the decoder and obtain the top1, top2, top3, top4 outputs using Equation 2. Results are shown in Figure 7(b) in this updated version. We can observe that i) the top1 curve is around the mean of the time series; ii) the top2 curve has similar shape to the ground truth (i.e., the input time series), which implies that using two memory units can already roughly represent its shape; iii) the top3 and top4 curves are even more refined, providing more details of the time series.

Table 9: MAE forecasting results on multivariate time series. Results of baselines marked with superscript $\diamond$ are copied from Wu et al. (2021); Zhou et al. (2022); Zeng et al. (2022). The best results are in bold, and the second best are underlined.

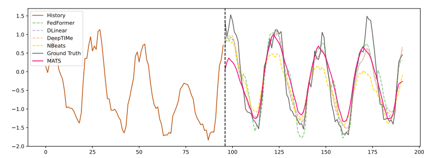
Datasets		Electricity	Exchange	Traffic	Weather	ETTh1	ETTh2	ETTm1	ETTm2		ILI
Methods	H	MAE	MAE	MAE	MAE	MAE	MAE	MAE	MAE	H	MAE
MATS	96	**0.270**	**0.128**	**0.307**	**0.161**	**0.382**	**0.319**	**0.289**	**0.223**	24	**0.727**
	192	**0.278**	**0.162**	**0.318**	**0.197**	**0.409**	**0.333**	**0.364**	**0.245**	36	**0.844**
	336	**0.284**	**0.203**	**0.320**	**0.218**	**0.422**	**0.363**	**0.388**	**0.271**	48	1.020
	720	**0.296**	**0.285**	**0.337**	**0.254**	**0.438**	**0.391**	**0.407**	**0.320**	60	**1.012**
VQLSTM	96	0.291	0.146	0.339	0.166	0.434	0.315	0.377	0.221	24	0.911
	192	0.293	0.184	0.341	0.211	0.456	0.346	0.415	0.250	36	0.899
	336	0.296	0.242	0.344	0.227	0.474	0.389	0.420	0.283	48	**0.966**
	720	0.307	0.449	0.353	0.279	0.485	0.471	0.458	0.345	60	1.122
FEDFormer$^\diamond$	96	0.308	0.278	0.366	0.296	0.419	0.388	0.419	0.287	24	1.260
	192	0.315	0.380	0.373	0.336	0.448	0.439	0.441	0.328	36	1.080
	336	0.329	0.500	0.383	0.380	0.465	0.487	0.459	0.366	48	1.078
	720	0.355	0.841	0.382	0.428	0.507	0.474	0.490	0.415	60	1.157
Autoformer$^\diamond$	96	0.317	0.323	0.388	0.336	0.459	0.397	0.475	0.339	24	1.287
	192	0.334	0.369	0.382	0.367	0.482	0.452	0.496	0.340	36	1.148
	336	0.338	0.524	0.337	0.395	0.496	0.486	0.537	0.372	48	1.085
	720	0.361	0.941	0.408	0.428	0.512	0.511	0.561	0.432	60	1.125
Informer$^\diamond$	96	0.368	0.752	0.391	0.384	0.713	1.525	0.571	0.453	24	1.677
	192	0.386	0.895	0.379	0.544	0.792	1.931	0.669	0.563	36	1.467
	336	0.394	1.036	0.420	0.523	0.809	1.835	0.871	0.887	48	1.469
	720	0.439	1.310	0.472	0.741	0.865	1.625	0.823	1.338	60	1.564
Pyraformer$^\diamond$	96	0.449	1.105	0.468	0.556	0.612	0.597	0.510	0.507	24	2.012
	192	0.443	1.151	0.467	0.624	0.681	0.683	0.537	0.673	36	2.031
	336	0.443	1.172	0.469	0.753	0.738	0.747	0.655	0.845	48	2.057
	720	0.445	1.206	0.473	0.934	0.782	0.783	0.724	1.451	60	2.100
LogTrans$^\diamond$	96	0.357	0.812	0.384	0.490	0.740	1.197	0.546	0.642	24	1.444
	192	0.368	0.851	0.390	0.589	0.824	1.635	0.700	0.757	36	1.467
	336	0.380	1.081	0.408	0.652	0.932	1.604	0.832	0.872	48	1.468
	720	0.376	1.127	0.396	0.675	0.852	1.540	0.820	1.328	60	1.560
Reformer$^\diamond$	96	0.402	0.829	0.423	0.596	0.728	1.317	0.528	0.619	24	1.382
	192	0.433	0.906	0.420	0.638	0.766	2.979	0.592	0.827	36	1.448
	336	0.433	0.976	0.420	0.596	0.835	2.769	0.721	0.972	48	1.465
	720	0.420	1.016	0.423	0.792	0.889	1.697	0.841	1.242	60	1.483
LSTMa	96	0.439	0.726	1.021	0.377	0.662	1.141	0.749	0.630	24	1.385
	192	0.459	0.922	0.968	0.493	0.753	1.026	0.777	1.073	36	1.435
	336	0.638	1.053	0.886	0.568	0.819	1.221	0.805	0.742	48	1.441
	720	0.870	1.227	0.868	0.562	0.893	1.286	0.800	1.239	60	1.464
LSTM$^\diamond$	96	0.437	1.049	0.453	0.406	0.675	1.221	0.939	1.073	24	1.734
	192	0.473	1.179	0.453	0.435	0.867	1.674	0.913	1.112	36	1.845
	336	0.473	1.231	0.455	0.454	0.994	1.549	1.124	1.238	48	1.857
	720	0.814	1.427	0.805	0.520	1.322	1.788	1.555	1.287	60	1.879
TCN	96	0.813	1.432	0.784	0.589	0.788	1.438	0.677	1.330	24	1.830
	192	0.821	1.444	0.794	0.600	0.765	1.440	0.690	1.339	36	1.879
	336	0.824	1.459	0.799	0.608	0.782	1.448	0.707	1.348	48	1.892
	720	0.784	1.458	0.804	0.610	0.794	1.481	0.733	1.354	60	1.918
LSTNet	96	0.645	1.058	0.685	0.587	0.960	1.687	1.215	1.365	24	1.770
	192	0.676	1.028	0.706	0.565	1.214	2.513	1.542	1.369	36	1.668
	336	0.727	1.031	0.730	0.587	1.369	2.591	2.076	1.369	48	1.787
	720	0.811	1.243	0.805	0.599	1.380	3.709	2.941	1.368	60	1.720
DLinear$^\diamond$	96	0.276	0.197	0.396	0.255	0.400	0.352	0.372	0.273	24	1.040
	192	0.280	0.292	0.370	0.296	0.432	0.352	0.389	0.325	36	1.088
	336	0.296	0.391	0.373	0.335	0.459	0.490	0.413	0.367	48	1.086
	720	0.329	0.574	0.394	0.381	0.516	0.538	0.453	0.421	60	1.146
DeepTIMe	96	0.280	0.196	0.397	0.257	0.406	0.380	0.368	0.282	24	1.017
	192	0.280	0.286	0.369	0.297	0.438	0.444	0.391	0.337	36	0.991
	336	0.297	0.372	0.371	0.336	0.462	0.486	0.413	0.380	48	1.101
	720	0.331	0.630	0.397	0.392	0.516	0.611	0.459	0.481	60	1.190
Repeat-C$^\diamond$	96	0.946	0.196	1.079	0.254	0.713	0.422	0.665	0.328	24	1.701
	192	0.950	0.289	1.087	0.292	0.733	0.473	0.690	0.371	36	1.884
	336	0.961	0.396	1.095	0.338	0.744	0.508	0.707	0.410	48	1.798
	720	0.975	0.681	1.097	0.394	0.756	0.517	0.729	0.465	60	1.677

Table 10: MAE forecasting results on univariate time series. Results of baselines marked with superscript $\diamond$ are copied from Wu et al. (2021); Zhou et al. (2022); Zeng et al. (2022). The best results are in bold, and the second best are underlined.

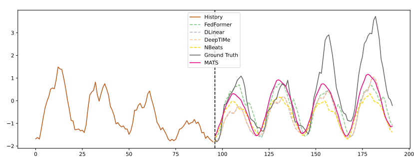
Datasets		Electricity	Exchange	Traffic	Weather	ETTh1	ETTh2	ETTm1	ETTm2		ILI
Methods	H	MAE	MAE	MAE	MAE	MAE	MAE	MAE	MAE	H	MAE
MATS	96	**0.306**	**0.106**	**0.230**	**0.012**	**0.123**	0.247	**0.074**	**0.094**	24	**0.527**
	192	**0.359**	**0.165**	**0.236**	**0.018**	**0.144**	**0.267**	**0.093**	**0.143**	36	**0.568**
	336	**0.365**	**0.212**	**0.248**	**0.020**	**0.166**	**0.314**	**0.112**	**0.180**	48	**0.590**
	720	**0.388**	**0.398**	**0.277**	**0.023**	**0.184**	**0.360**	**0.139**	**0.215**	60	**0.588**
VQLSTM	96	0.435	0.111	0.350	0.014	0.134	**0.227**	0.082	0.122	24	0.712
	192	0.451	0.186	0.341	0.021	0.159	0.271	0.103	0.159	36	0.791
	336	0.475	0.276	0.335	0.024	0.171	0.337	0.124	0.197	48	0.848
	720	0.500	0.429	0.342	0.027	0.199	0.362	0.162	0.244	60	0.772
FEDFormer$^{\diamond}$	96	0.370	0.304	0.312	0.062	0.215	0.271	0.140	0.198	24	0.627
	192	0.386	0.420	0.312	0.062	0.245	0.330	0.186	0.245	36	0.617
	336	0.431	0.555	0.323	0.050	0.270	0.378	0.231	0.279	48	0.697
	720	0.484	0.879	0.344	0.059	0.299	0.420	0.250	0.325	60	0.774
Autoformer$^{\diamond}$	96	0.438	0.387	0.346	0.081	0.206	0.306	0.183	0.189	24	0.732
	192	0.428	0.369	0.370	0.067	0.262	0.351	0.216	0.256	36	0.650
	336	0.470	0.524	0.371	0.062	0.258	0.389	0.218	0.305	48	0.752
	720	0.581	0.867	0.372	0.070	0.283	0.409	0.267	0.335	60	0.797
Informer$^{\diamond}$	96	0.367	0.944	0.353	0.044	0.377	0.373	0.277	0.225	24	2.050
	192	0.388	0.924	0.376	0.040	0.395	0.387	0.310	0.283	36	1.916
	336	0.423	1.296	0.387	0.049	0.381	0.401	0.591	0.336	48	1.846
	720	0.599	0.953	0.436	0.042	0.355	0.439	0.586	0.435	60	2.057
Pyraformer$^{\diamond}$	96	0.381	0.368	0.254	0.068	0.424	0.303	0.224	0.208	24	1.988
	192	0.389	0.663	0.256	0.063	0.440	0.329	0.347	0.261	36	1.785
	336	0.416	1.297	0.264	0.065	0.461	0.394	0.359	0.325	48	1.885
	720	0.551	1.328	0.319	0.068	0.485	0.376	0.533	0.372	60	1.992
LogTrans$^{\diamond}$	96	0.393	0.377	0.004	0.052	0.468	0.379	0.171	0.208	24	1.662
	192	0.483	0.619	0.006	0.060	0.409	0.429	0.317	0.275	36	1.363
	336	0.483	1.070	0.006	0.054	0.546	0.437	0.459	0.302	48	1.575
	720	0.531	1.175	0.007	0.059	0.628	0.387	0.579	0.321	60	1.733
Reformer$^{\diamond}$	96	0.379	0.444	0.383	0.087	0.569	0.838	0.355	0.214	24	0.083
	192	0.402	0.719	0.453	0.044	0.575	1.671	0.474	0.290	36	1.520
	336	0.448	1.128	0.468	0.100	0.589	1.582	0.583	0.312	48	1.749
	720	0.511	0.956	0.433	1.720	0.666	1.039	0.730	0.335	60	1.847
LSTMa	96	0.731	0.879	1.640	0.068	0.601	0.609	0.597	0.720	24	1.229
	192	0.734	1.347	1.531	0.104	0.878	0.846	0.863	0.863	36	1.443
	336	0.898	1.435	1.442	0.090	1.056	0.998	1.114	0.931	48	1.524
	720	0.966	1.570	1.438	0.106	1.236	1.070	1.234	1.044	60	1.511
DLinear$^{\diamond}$	96	0.438	0.241	0.398	0.057	0.184	0.273	0.136	0.438	24	0.920
	192	0.423	0.358	0.346	0.061	0.212	0.328	0.194	0.423	36	0.939
	336	0.443	0.465	0.342	0.065	0.249	0.379	0.210	0.443	48	1.013
	720	0.481	0.700	0.379	0.069	0.344	0.461	0.242	0.481	60	1.199
DeepTIMe	96	0.444	0.222	0.393	0.048	0.182	0.277	0.136	0.196	24	0.798
	192	0.431	0.332	0.332	0.028	0.216	0.334	0.167	0.243	36	0.859
	336	0.449	0.440	0.324	0.031	0.257	0.382	0.200	0.284	48	0.943
	720	0.491	0.633	0.362	0.032	0.443	0.482	0.256	0.337	60	0.983
N-BEATS	96	0.409	0.299	0.243	0.039	0.257	0.286	0.159	0.191	24	1.116
	192	0.431	0.665	0.249	0.043	0.419	0.320	0.378	0.260	36	0.949
	336	0.450	0.605	0.254	0.045	0.285	0.390	0.433	0.315	48	1.029
	720	0.501	0.860	0.295	0.053	0.419	0.458	0.317	0.359	60	1.251
ARIMA	96	0.887	0.245	0.863	0.035	0.215	0.384	0.176	0.857	24	1.015
	192	0.926	0.404	0.975	0.052	0.270	0.467	0.258	1.535	36	1.047
	336	1.015	0.598	1.053	0.078	0.340	0.579	0.375	2.554	48	1.100
	720	1.227	0.935	1.115	0.155	0.396	0.777	0.689	5.312	60	1.205
Repeat-C$^{\diamond}$	96	0.997	0.221	1.532	0.025	0.203	0.423	0.136	0.354	24	0.907
	192	0.998	0.435	1.527	0.028	0.236	0.462	0.169	0.387	36	0.909
	336	1.029	0.468	1.525	0.031	0.266	0.502	0.196	0.414	48	0.921
	720	1.058	0.764	1.515	0.036	0.284	0.532	0.232	0.457	60	0.959



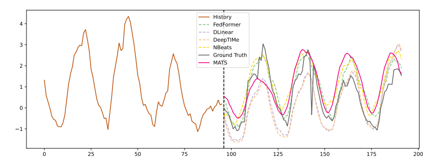
(a) 1



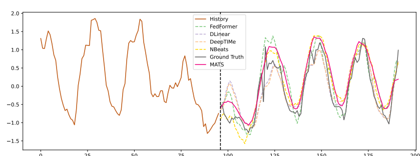
(b) 161



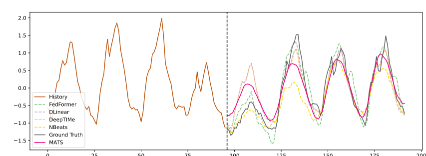
(c) 321



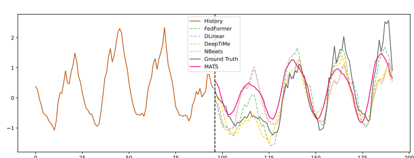
(d) 481



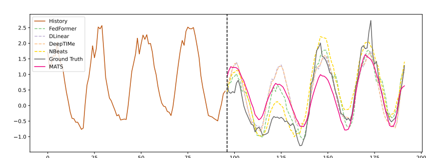
(e) 641



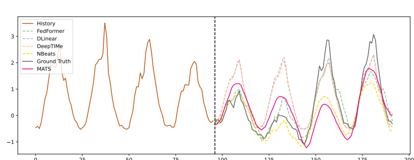
(f) 801



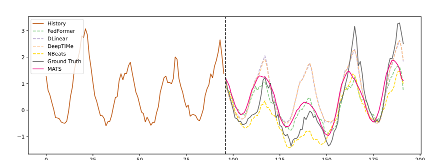
(g) 961



(h) 1121



(i) 1281

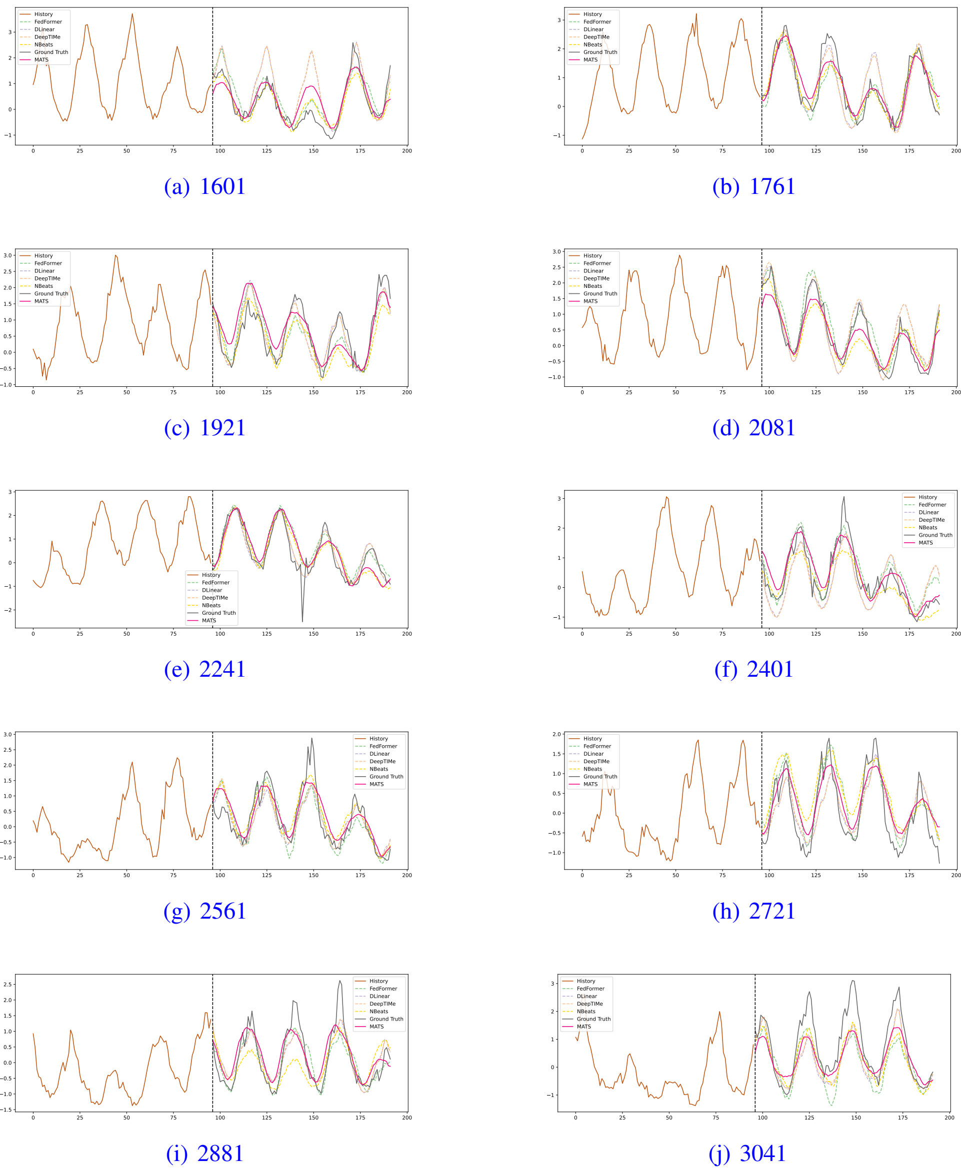


(j) 1441

Figure 5: First 10 test samples visualization on univariate Electricity.

Table 11: MSE with or without discriminator on the univariate Electricity and Exchange.

	H	Univariate Electricity	Univariate Exchange	Multivariate Electricity	Multivariate Exchange
With Discriminator	96	**0.177**	**0.019**	**0.156**	**0.034**
	192	0.252	**0.044**	**0.165**	0.049
	336	**0.260**	**0.069**	**0.168**	**0.073**
	720	**0.286**	0.262	**0.179**	0.135
Without Discriminator	96	0.185	**0.019**	0.158	0.034
	192	**0.243**	0.045	**0.165**	0.048
	336	0.277	0.075	0.169	**0.073**
	720	0.311	**0.219**	0.183	**0.126**



(a) 1601 (b) 1761

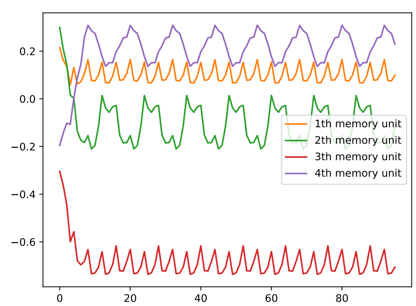
(c) 1921 (d) 2081

(e) 2241 (f) 2401

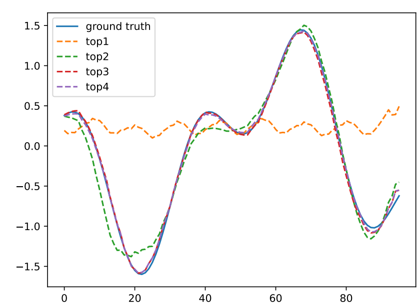
(g) 2561 (h) 2721

(i) 2881 (j) 3041

Figure 6: Last 10 samples visualization on univariate Electricity.



(a) Decoded results to the 4 memory units of the toy dataset.



(b) Decoded results to the topK memory attentions of the toy dataset.

Figure 7: Visualization regarding memory attentions and memory units of the toy dataset.