TARNet: Task-Aware Reconstruction for Time-Series Transformer

Ranak Roy Chowdhury
rrchowdh@eng.ucsd.edu
University of California, San Diego
La Jolla, CA, USA

Xiyuan Zhang
xiyuanzh@ucsd.edu
University of California, San Diego
La Jolla, CA, USA

Jingbo Shang*
jshang@eng.ucsd.edu
University of California, San Diego
La Jolla, CA, USA

Rajesh K. Gupta*
gupta@eng.ucsd.edu
University of California, San Diego
La Jolla, CA, USA

Dezhi Hong*†
hondezhi@amazon.com
Amazon

ABSTRACT

Time-series data contains temporal order information that can guide representation learning for predictive end tasks (e.g., classification, regression). Recently, there are some attempts to leverage such order information to first pre-train time-series models by reconstructing time-series values of randomly masked time segments, followed by an end-task fine-tuning on the same dataset, demonstrating improved end-task performance. However, this learning paradigm decouples data reconstruction from the end task. We argue that the representations learnt in this way are not informed by the end task and may, therefore, be sub-optimal for the end-task performance. In fact, the importance of different timestamps can vary significantly in different end tasks. We believe that representations learnt by reconstructing *important* timestamps would be a better strategy for improving end-task performance. In this work, we propose TARNet[1], **T**ask-**A**ware **R**econstruction **Net**work, a new model using Transformers to learn task-aware data reconstruction that augments end-task performance. Specifically, we design a data-driven masking strategy that uses self-attention score distribution from end-task training to sample timestamps deemed important by the end task. Then, we mask out data at those timestamps and reconstruct them, thereby making the reconstruction task-aware. This reconstruction task is trained alternately with the end task at every epoch, sharing parameters in a single model, allowing the representation learnt through reconstruction to improve end-task performance. Extensive experiments on tens of classification and regression datasets show that TARNet significantly outperforms state-of-the-art baseline models across all evaluation metrics.

CCS CONCEPTS

• **Mathematics of computing → Time series analysis**; • **Computing methodologies → Supervised learning by classification**; **Supervised learning by regression**.

[1]Code is publicly available at https://github.com/ranakroychowdhury/TARNet

*Corresponding authors.
†Work unrelated to Amazon.

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08 .
https://doi.org/10.1145/3534678.3539329

KEYWORDS

Time series; self-supervision; data reconstruction; self-attention

ACM Reference Format:
Ranak Roy Chowdhury, Xiyuan Zhang, Jingbo Shang*, Rajesh K. Gupta*, and Dezhi Hong*†. 2022. TARNet: Task-Aware Reconstruction for Time-Series Transformer. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3534678.3539329

1 INTRODUCTION

Time-series data has domain-specific structural properties encoded in the temporal ordering of events. These intrinsic properties can provide a rich source of supervision besides target labels, which the state-of-the-art time-series models [2, 38] often neglect. Recently, time-series Transformer [37] leveraged this unlabeled data to craft a reconstruction task that masks time-series values of randomly chosen time segments and reconstructs them. The pre-trained model is then fine-tuned on an end task, by reusing the *same* data samples along with their labels, leading to improved performance over exclusively doing supervised learning on the end task.

However, this data reconstruction task precedes fine-tuning as a decoupled step, which means the representation learnt during reconstruction is not informed about the end task. Hence, such learnt representation may not be fully leveraged to perform optimally on the end task.

Depending on the end task, different properties of the given data may be useful for different end tasks. For example, consider the following end tasks using the same data collected from sensors in a building: predict the level of energy consumption (high, medium, low) and the occupancy status (occupied or not) of a room based on outdoor temperature and humidity, and light intensity and CO_2 readings from a room. Energy consumption prediction task may be highly correlated to times when temperature is high (air conditioning stays on) or light intensity is high (lights are switched ON) while occupancy status may correlate to timestamps when CO_2 level is high. Hence, depending on the end task, certain timestamps in the data may be more important than others for that task.

Generic learnt representations typically result from decoupled data reconstruction and end tasks. To optimize the performance for an end task, we customize the learnt representation for the end task in TARNet. We test and validate the hypothesis that a representation learnt by reconstructing data from timestamps important to the end task will yield improved performance over reconstruction

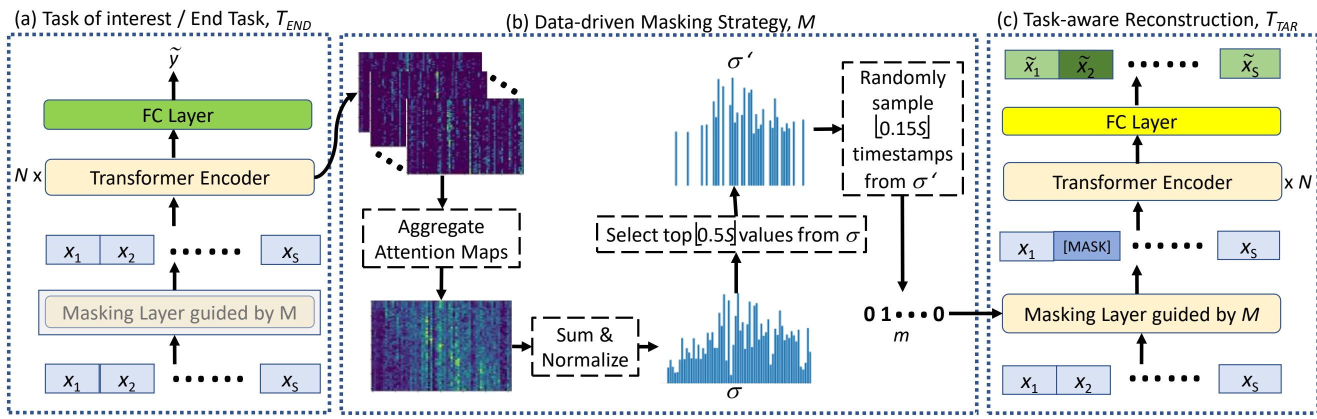


Figure 1: TARNet Overview: (a) Task of interest / End Task, T_{END}: Data is mean-standardized, then passed through an Embedding and a Positional Encoding layer (not shown for simplicity), followed by the N-layer Transformer Encoders and Fully Connected (FC) Layer; (b) Data-driven Masking Strategy, M: For every time-series data, we collect attention maps generated by Transformer Encoders in T_{END} and then compute the set of important timestamps to be masked in task-aware reconstruction; and (c) Task-aware Reconstruction, T_{TAR}: Input data are masked at timestamps computed by M and reconstructed. Transformer Encoder parameters are shared between T_{END} and T_{TAR}, but the FC layers are different (highlighted by different colors).

on random time segments. Therefore, we design a data reconstruction task which masks data from those important timestamps and reconstructs them. In the process, the model learns a task-specific representation, resulting in improved end task performance.

Figure 1 shows TARNet's learning process. Using a transformer encoder [29] as the backbone model, we train for the end task (Figure 1(a)) and the data reconstruction task (Figure 1(c)) alternately on the same model. In order to compute the timestamps to mask during data reconstruction, we design a data-driven masking strategy (Figure 1(b)). It uses the self-attention score distribution generated by transformer encoder during the end task training and determines the set of timestamps to mask. Since the two tasks share parameters, the representation learnt during reconstruction can be effectively leveraged by the end task to improve performance.

We conducted experiments on 34 classification datasets from UEA Archive [1], UCI Machine Learning Repository [10, 15] and 6 regression datasets from Monash University, UEA, UCR Time Series Regression Archive [27]. Time Series Transformer (TST) [37], the current state-of-the-art for time-series, achieved the best accuracy on 6 out of 10 datasets, when compared with 5 baselines. We compared TARNet with 14 state-of-the-art baselines and it performed the best on 17 out of 34 datasets, being 2.7% higher in average accuracy than TST, which now performs best on 7 datasets. Similarly, TST achieved the lowest error on 3 out of 6 datasets for regression when compared with 11 state-of-the-art baselines. TARNet achieved the lowest error on 3 and 2nd lowest error on 2 datasets when compared with the same baselines, whereas TST now achieves the lowest error on 2 and 2nd lowest error on 1 dataset. We conducted case studies to show how TARNet's data-driven masking strategy learns task-specific representations, consistent with domain characteristics, thereby boosting end-task performance.

In summary, our main contributions are:

- We propose TARNet to learn task-aware reconstruction from time-series data to augment end-task performance.
- We design a data-driven masking strategy to determine important timestamps to an end task and learn to reconstruct them.
- We evaluate TARNet on numerous real-world datasets to validate and quantify its efficacy compared with state-of-the-art methods.

2 RELATED WORK

2.1 Non-Deep Learning Methods

ROCKET [5] and MiniROCKET [6] recently produced state-of-the art results for time-series. They learn features extracted by numerous and various random convolutional kernels. Other relevant directions include: (1) time series shapelet, (2) bag-of-patterns, and (3) distance-based models. Baydogan et al. [3] introduced Symbolic Representation to learn local relationships between different dimensions. Shapelets [33] are short discriminative time series sub-sequences, e.g. dynamic shapelets [23], efficient shapelets [16]. WEASEL-MUSE [24] utilizes bag of SFA (Symbolic Fourier Approximation). Distance-based methods [8, 31] use distance metric to measure similarity of a pair of time series. Among limitations of these approaches are that they incorporate expert insights, consist of large, heterogeneous ensembles of classifiers, scale poorly to long time-series, and many apply to only uni-variate time-series.

TARNet can be applied to both uni- and multi-variate time-series, automatically extracts features, and handles long time-series.

2.2 Deep Learning Methods

Using labeled data. Fawaz et al. [12] summarize many neural networks-based methods for time-series. Most neural networks-based methods use some arrangement of LSTM, CNN or both [18, 39]. Others use different components of neural models, e.g., learnable temporal pooling [19], correlative channel-aware learnable fusion [2], label-learning [22], attentional prototype network [38], and shapelet embedding [20]. TARNet proposes a subsidiary data reconstruction technique that utilizes knowledge from the end

task to learn a task-specific data representation. Sharing parameters of this reconstruction task with the end task in a single architecture allows the learnt representation to improve end task performance.
Using both unlabeled and labeled data. Unsupervised representation learning for time-series uses triplet loss with negative sampling [14], hierarchical contrastive loss [36], temporal and contextual contrasting [11], local smoothness to define neighborhoods in time [28], and reprogramming acoustic models [32]. TST [37] first pre-trains a transformer model by an unsupervised objective; masks out time-series values at random time segments from data and reconstructs them. It then reuses the *same* training samples to fine-tune the model on an end task. This gave improved performance than using the data once to train a fully supervised model.

However, decoupling the data reconstruction from the end task makes the representation learnt during reconstruction uninformed about the end task. Depending on the end task, certain timestamps in time-series data may be more important than others [21], which the learnt representation ignores. TARNet aims to learn a task-aware data reconstruction by masking important timestamps with respect to the end task. Hence, the learnt representation is better suited for improving end task performance than the representation learnt from reconstructing randomly masked time segments.

3 TARNET

In Figure 1, we show a schematic diagram of TARNet common across all considered tasks. In this section, we first present the problem setting and base model architecture shared by the two tasks. Then, we explain the end-task T_{END} (i.e., Figure 1(a)) and task-aware reconstruction T_{TAR} (i.e, Figure 1(c)). Finally, we present our data-driven masking strategy (i.e., Figure 1(b)) that uses information from T_{END} to decide which timestamps to mask for T_{TAR}.

3.1 Problem Description and Notations

Each training sample $\mathrm{X} \in \mathbb{R}^{S \times N}$ denotes a multivariate time-series of length S and N variables. Specifically, it comprises a sequence of S N-dimensional feature vectors, $x_t \in \mathbb{R}^N : \mathrm{X} \in \mathbb{R}^{S \times N}$. This formulation also covers the uni-variate case when $N = 1$. All the training samples come together with a target label y, which is an integer class id for a classification task or a real-valued number for a regression task. The full training dataset is labeled, i.e. we do not leverage any additional unlabeled data. Based on these training samples, we build a model to predict the label $\tilde{y}$ of unseen data X.

3.2 Base Model

We opt to use Transformer Encoders [29] as the backbone model, as we aim to develop a general framework to learn task-specific reconstruction that can be applied for a multitude of tasks. An architecture consisting of an encoder provides flexibility as it can not only handle tasks like classification, regression, imputation, but also handle generative tasks such as forecasting. One can plug in a task of interest by replacing the Fully Connected (FC) Layer in Figure 1(a) by task-specific layers (e.g., decoder for forecasting).

The feature vectors x_t are first mean-standardized per variable dimension. Then x_t is linearly projected onto a D-dimensional vector space, where D is the dimension of the Transformer model sequence element representations (typically called embedding dimension):

$$u_t = \mathrm{W}_p x_t + b_p, \tag{1}$$

where $\mathrm{W}_p \in \mathbb{R}^{D \times N}$, $b_p \in \mathbb{R}^D$ are learnable parameters and $u_t \in \mathbb{R}^D, t = 1, 2, ..., S$ are the model input vectors. The Transformer is a feed-forward architecture insensitive to the ordering of input. Therefore, we add positional encoding to these input vectors in order to make it aware of the sequential nature of the time series. The resultant vectors become the queries, keys and values of the self-attention layer in the encoder block. We pass data through several layers of such Transformer encoder blocks. Then, we pass the output values weighted by self-attention scores through a fully connected feed-forward network. We refer the reader to the original work [29] for a detailed description of the Transformer model.

3.3 End Task (T_{END})

For clarity, we use classification and regression as example end tasks here. Please note that TARNet can be easily extended to other tasks such as anomaly detection and time-series forecasting, by tweaking the FC Layer in Figure 1(a).

We modify the base model architecture presented in Section 3.2 for regression and classification in the following way:

The data fed to T_{END} is not masked, as illustrated by the frozen Masking Layer in Figure 1(a). The vector corresponding to the last timestamp from Transformer Encoders $z_t \in \mathbb{R}^D$ is fed through 2 FC layers and *RELU* activation (represented as f), with parameters

$$\mathrm{W}_{L1} \in \mathbb{R}^{K_E \times D}, b_{L1} \in \mathbb{R}^{K_E}, \mathrm{W}_{L2} \in \mathbb{R}^{K_E \times K_E}, b_{L2} \in \mathbb{R}^{K_E},$$

followed by the output layer with parameters

$$\mathrm{W}_E^O \in \mathbb{R}^{C \times K_E}, b_E^O \in \mathbb{R}^C,$$

where K_E is the feed-forward dimension of FC Layer for T_{END} and C is the number of classes for classification or number of scalars to be estimated for regression (typically $C = 1$):

$$\tilde{y} = \mathrm{W}_E^O f(\mathrm{W}_{L2} f(\mathrm{W}_{L1} z_t + b_{L1}) + b_{L2}) + b_E^O. \tag{2}$$

For classification, predictions $\tilde{y}$ are passed through a softmax to give probability distribution, p, over C classes. We use cross-entropy loss with categorical ground truth labels, $\mathcal{L}_{END} = \sum_{i=1}^{C} y_i log(p_i)$. For regression, we use squared error, $\mathcal{L}_{END} = \|\tilde{y} - y\|_2^2$.

3.4 Task-aware Reconstruction (T_{TAR})

Learning data representation through reconstruction has been explored in natural language processing [7] and time-series [37]. The goal of T_{TAR}, illustrated in Figure 1(c), is to learn a data representation by reconstructing the input data X after it has been appropriately masked by the Data-driven Masking Strategy, M.

The role of TARNet's masking strategy M, elaborated in Section 3.5, is to generate a new binary training data mask $m \in \mathbb{R}^S$ for each training sample at every epoch. It is a boolean array with $\lfloor \mu S \rfloor$ number of 1's, where μ is a hyper-parameter $0 < \mu < 1$, to select the timestamps to be masked from X for the reconstruction task. Let m_t represent the value of m at timestamp t. If $m_t = 1$ we mask x_t, otherwise we do not. Masking a particular timestamp, t, involves replacing the N-dimensional feature vector x_t with zeros. X passes through Transformer Encoder layers after being masked by m. The final representation vectors $Z \in \mathbb{R}^{S \times D}$ is fed through 2 FC layers and *RELU* activation, with parameters

$$W_{L3} \in \mathbb{R}^{K_R \times D}, b_{L3} \in \mathbb{R}^{K_R}, W_{L4} \in \mathbb{R}^{K_R \times K_R}, b_{L4} \in \mathbb{R}^{K_R},$$

followed by the output layer with parameters

$$W_R^O \in \mathbb{R}^{N \times K_R}, b_R^O \in \mathbb{R}^N,$$

where K_R is the feed-forward dimension of FC Layer for T_{TAR} and N is the number of variables:

$$\tilde{X} = W_R^O f(W_{L4} f(W_{L3} Z + b_{L3}) + b_{L4}) + b_R^O. \tag{3}$$

The label for this task is the raw input data X. To ensure accurate reconstruction, we calculate Mean Square Error (MSE) between the ground truth X and prediction $\tilde{X}$. We calculate the average MSE loss for masked and unmasked part of the data as follows:

$$\mathcal{L}_{masked} = \frac{1}{N \sum_{t=1}^{S} m_t} \sum_{t=1}^{S} m_t \|\tilde{x_t} - x_t\|_2^2, \tag{4}$$

$$\mathcal{L}_{unmasked} = \frac{1}{N(S - \sum_{t=1}^{S} m_t)} \sum_{t=1}^{S} (1 - m_t) \|\tilde{x_t} - x_t\|_2^2. \tag{5}$$

Unlike TST, which only considers MSE loss for reconstructing the masked portion of the data, $\mathcal{L}_{masked}$, we include loss incurred for replicating the unmasked, observed portion of the input data, $\mathcal{L}_{unmasked}$, as well. Time-series data is auto-regressive with strong correlation across time. Therefore, the ability to reconstruct the masked data at a given timestamp depends on how effectively the model learns to reconstruct the unmasked data and use that as context to infer the masked data. Including the loss for the unmasked data ensures its accurate reconstruction.

The combined reconstruction loss $\mathcal{L}_{TAR}$ is a weighted sum of $\mathcal{L}_{masked}$ and $\mathcal{L}_{unmasked}$, given by

$$\mathcal{L}_{TAR} = \lambda \mathcal{L}_{masked} + (1 - \lambda) \mathcal{L}_{unmasked}, \tag{6}$$

where λ is a hyper-parameter $0 < \lambda < 1$ that controls the relative weights between the two losses. It is advisable to keep $\lambda > 0.5$ because the masked timestamps are more important for the end task than the unmasked ones.

With $\mathcal{L}_{END}$ as the end task loss, the total loss becomes

$$\mathcal{L}_{Total} = \eta \mathcal{L}_{TAR} + (1 - \eta) \mathcal{L}_{END}, \tag{7}$$

where η is a hyper-parameter $(0 < \eta < 1)$ that controls the relative weights between the two task losses. We train T_{END} and T_{TAR} end-to-end alternately at every epoch, until convergence.

3.5 Data-driven Masking Strategy (M)

Data reconstruction in Time-series Transformer [37] involves masking segment of time-series data at randomly chosen timestamps and reconstructing them. However, different timestamps in the data may have different levels of importance to the end task. Therefore, we eschew random reconstruction of data in favor of a strategy that uses end task characteristics. Specifically, we identify timestamps that the end task deemed *important* during learning. We will then mask x_t from X corresponding to those timestamps and reconstruct them during T_{TAR}. We hypothesize that reconstructing data at timestamps identified to be important by the end task will generate a data representation that benefits the end task. This is in contrast to a random masking based data reconstruction, which does not consider any such information.

Algorithm 1 Training of TARNet

Input: X, y
Hyper-parameters: $\mu, \beta, \lambda, \eta$
Output: *Model*

σ initialized randomly
Model = TransformerEncoder()
while training **do**
  σ' = top $\lfloor \beta S \rfloor$ values from σ
  $m \sim$ Randomly sample $\lfloor \mu S \rfloor$ timestamps without replacement from σ'
  $\tilde{X}, \tilde{y}$, A = *Model*.train(X, m) # A ← Self-Attention Scores
  Compute $\mathcal{L}_{TAR}(\tilde{X}, X, \lambda)$ and $\mathcal{L}_{END}(\tilde{y}, y)$
  $\mathcal{L}_{Total} = \eta \mathcal{L}_{TAR} + (1 - \eta) \mathcal{L}_{END}$
  σ = add_and_normalize(A)
end while
return *Model*

To define the notion of an "important" timestamp, we use self-attention weights generated by Transformer Encoder in the forward pass of T_{END}. Attention weights indicate how much weight should be assigned to each x_t to compute representation for a given x_t. We compute aggregate attention map $A \in \mathbb{R}^{S \times S}$ by summing the attention maps generated by each layer of Transformer Encoder. Let A_{ik} be the attention weight assigned to x_k during update of x_i, where $i = k = 1, 2, ..., S$, and $\sum_{k=1}^{S} A_{ik} = 1$ for all i. Therefore, the update to x_i is a weighted sum of $x_{1,2,...,S}$, where the weights are $A_{i,k=1,2,...S}$. We compute $\sigma \in \mathbb{R}^S$, where $\sigma_k = \frac{\sum_{i=1}^{S} A_{ik}}{\sum_{k=1}^{S} \sum_{i=1}^{S} A_{ik}}$ for $k = 1, 2, ..., S$. σ_k represents the normalized aggregate attention weight of timestamp k to the computation of $x_1, x_2, ..., x_S$. We define the importance of each timestamp by its magnitude in σ, i.e. the higher σ_k is, the more important timestamp k is for T_{END}.

We then select the timestamps corresponding to the top $\lfloor \mu S \rfloor$ values in σ and mask them from X for reconstruction. Since the same training data is fed at every epoch, the set of important timestamps computed from a given sample will not vary across epochs. Hence, the model may memorize reconstructing a few selected timestamps from the sample, leading to *overfitting*. Considering the heterogeneity in time-series data due to irregular sampling frequency or uncertainty about feature availability, it is probable that real-world data may have a different set of important timestamps compared to those seen in training data. Therefore, not exploring enough timestamps to approximate the training data distribution may lead to poor generalization on the real-world data.

Hence, we ensure that for every sample, at each epoch the model explores a random set of timestamps among those that are important. Therefore, we introduce an attention regularization parameter, β, where $\beta > \mu$ and $0 < \beta < 1$. We, therefore, compute set σ' to choose the top $\lfloor \beta S \rfloor$ values in σ. Then we randomly sample $\lfloor \mu S \rfloor$ timestamps without replacement from σ' to generate the training data mask m. $m_t = 1$ if t is sampled from σ', otherwise 0.

Although we still choose an important set of timestamps to mask, the use of randomization through sampling ensures that the model does not always mask the same set of timestamps for a sample throughout its entire training regime. This gives the model a more versatile representation of the underlying data distribution, yet, one that is important for the end task. This data-driven masking strategy makes the model learn task-specific data representation by reconstructing data at those timestamps deemed important by the end task. Algorithm 1 outlines the training procedure of TARNet.

4 EXPERIMENTS

We present the datasets, baselines, training settings, followed by the evaluation metrics. We then show and analyze classification and regression results of TARNet. We also conduct an ablation study, few-shot training experiments and case studies to justify TARNet.

4.1 Experimental Setup

We use benchmark time-series datasets with detailed information available in UEA Archive [1], UCI Machine Learning Repository [10, 15], and Monash University, UEA, UCR Time Series Regression Archive [27]. These datasets represent an assortment of domains (Motion, Audio, EEG, HAR), sensor type, and sampling frequency. The number of training data points varies from 15 to over one million, the length of the time series, S, varies between 8 to 17,984, the number of features, N, varies between 1 to 1,345, and the number of target classes, C, varies between 2 to 39. $N = 1$ covers the uni-variate case. $N > 1$ refers to the multi-variate case.

We compare TARNet with statistical [1, 4–6, 9, 24–26] and deep learning [11–14, 18, 20, 28, 30, 35, 37, 38] baselines.

4.1.1 *Statistical Baselines.*

(1) **Distance-based method** [1]. Euclidean Distance (ED), dimension independent dynamic time warping (DTWI), and dimension-dependent dynamic time warping (DTWD) [25].
(2) **SVR**: [9] Support Vector Regression.
(3) **Tree-based methods**: Random Forest [26] and XGBoost [4].
(4) **WEASEL-MUSE** [24] is a bag-of-pattern based sliding-window approach with statistical feature extraction and filtration.
(5) **Rocket** [5] convolves time series with random convolutional kernels and applies global max pooling to extract features.
(6) **MiniRocket** [6] upgrades Rocket by speeding it up, using a small, fixed set of kernels, and is almost entirely deterministic.

4.1.2 *Deep Learning Baselines.*

(1) **FCN** [30] Fully Convolutional Networks. Replaces traditional final FC layer with a Global Average Pooling (GAP) layer.
(2) **MLSTM-FCNs** [18] expands LSTM-FCN and Attention LSTM-FCN by adding squeeze-and-excitation blocks.
(3) **Negative samples (NS)** [14] generates negative samples and trains a dilated causal convolution encoder with triplet loss.
(4) **TapNet** [38] designs random group permutation method with multi-layer convolutional and attentional prototype network.
(5) **ShapeNet** [20] extends shapelet [33] for multivariate time-series. Learns shared embedding space across different shapelet candidates, trains a dilated causal CNN, followed by an SVM.
(6) **Time Series Transformer (TST)** [37] pre-trains Transformer Encoder by masking random time segments and reconstructing them. Reuses the same data to fine-tune the model.
(7) **TS2Vec** [35] performs hierarchical contrastive learning over augmented context views. Builds representation of an arbitrary sub-sequence by aggregating representations of timestamps.
(8) **TNC** [28] leverages local smoothness of a signal to define temporal neighborhoods and learns generalizable representations.
(9) **TS-TCC** [11] encourages consistency of different data augmentations to learn transformation-invariant representations.
(10) **ResNet** [12] uses convolutional followed by a GAP layer. Adds shortcut residual connection between convolutional layers.
(11) **Inception** [13] is an ensemble of deep CNN models, inspired by the Inception-v4 architecture.

We normalize the datasets for each of our experiments. For datasets on which the accuracies of the baselines have been reported, we present the same results according to their papers. For the remaining datasets, we train all the baseline models with sufficient hyper-parameter tuning to produce results. Since our benchmark datasets are widely heterogeneous in terms of number of data points, features, sequence length, and sampling frequency, as well as the physical nature of the data itself, we obtain better performance via cursory tuning of architecture-specific hyper-parameters. To select hyper-parameters, we do a random 80%-20% split of the training set and used the 20% as a validation set for hyper-parameter tuning. After fixing the hyper-parameters, we train the model again using the entire training set and save the model with the lowest training loss. We use the saved model to evaluate on the official test set and report our evaluation metrics.

4.2 Evaluation Metrics

We use accuracy and Root Mean Squared Error (RMSE) error as our performance metric for classification and regression, respectively. Considering the large number of datasets and baselines used, it is highly unlikely for a single model to outperform all other methods on every datasets. Therefore, we also present some summary statistics to present a holistic and a fairer comparison of the methods. The evaluation metrics are as follows:

- **Ours 1-to-1 Wins/Draws/Losses**: Number of datasets for which TARNet's accuracy or RMSE is better/same/worse than the corresponding baselines, respectively. Higher wins, lower draws and lower losses are better. This is useful to draw a one-on-one comparison between TARNet and a given model.
- **Mean Rank**: Average rank of a model across all datasets. Lowest rank is assigned to model with highest accuracy for classification and lowest RMSE for regression. Lower mean rank is better.
- **Avg.Rel.Diff.Mean** [37]: We report the "average relative difference from mean" metric r_j for each model j, over N datasets:

$$r_j = \frac{1}{N} \sum_{i=1}^{N} \frac{R(i,j) - \bar{R}_i}{\bar{R}_i}, \qquad \bar{R}_i = \frac{1}{M} \sum_{j=1}^{M} R(i,j), \tag{8}$$

where $R(i, j)$ is the RMSE of model j on dataset i and M is the number of models. $r_j = -0.3$ means that the model on average attains 30% lower RMSE on a dataset than the average model performance on the same dataset. Lower value is better.

4.3 Classification

Table 1 shows the accuracy of the models. According to Table 1, the overall accuracy of TARNet is the best among all compared methods. TARNet performs the best on 17 datasets, as compared to 7 and 6 by the next best baselines TST [37] and Rocket [5], respectively. TARNet achieves a 2.7-point higher average accuracy across all

Table 1: Accuracy of TARNet and baselines on classification datasets from UEA Archive and UCI Machine Learning Repository. We mark the best and second best values. Baselines are presented in ascending order (left to right) by average accuracy. A dash indicates that the corresponding method failed to run on this dataset. Higher Total best accuracy, average accuracy, and Ours 1-to-1 Wins is better. Lower Ours 1-to-1 Draws, Ours 1-to-1 Losses, and Mean Rank is better.

Dataset	ED	MLSTM-FCNs	DTWD	TapNet	DTWI	NS	WEASEL-MUSE	TS-TCC	TNC	ShapeNet	TS2Vec	Rocket	MiniRocket	TST	TARNet
ArticularyWordRecognition	0.970	0.973	0.987	0.987	0.980	0.987	0.990	0.953	0.973	0.987	0.987	**0.993**	**0.993**	0.947	0.977
AtrialFibrillation	0.267	0.267	0.220	0.333	0.267	0.133	0.333	0.267	0.133	0.400	0.200	0.067	0.133	0.533	**1.000**
BasicMotions	0.676	0.950	0.975	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.975	**1.000**	0.975	**1.000**	**1.000**	0.925	**1.000**
CharacterTrajectories	0.964	0.985	0.989	**0.997**	0.969	0.994	0.990	0.985	0.967	0.980	0.995	0.991	0.990	0.971	0.994
Cricket	0.944	0.917	**1.000**	0.958	0.986	0.986	**1.000**	0.917	0.958	0.986	0.972	**1.000**	0.986	0.847	**1.000**
DuckDuckGeese	0.275	0.675	0.600	0.575	0.550	0.675	0.575	0.380	0.460	0.725	0.680	0.500	**0.750**	0.300	**0.750**
EigenWorms	0.549	0.504	0.618	0.489	-	0.878	**0.890**	0.779	0.840	0.878	0.847	0.650	0.790	0.720	0.420
Epilepsy	0.666	0.761	0.964	0.971	0.978	0.957	**1.000**	0.957	0.957	0.987	0.964	0.986	**1.000**	0.775	**1.000**
ERing	0.133	0.133	0.133	0.133	0.133	0.133	0.133	0.904	0.852	0.133	0.874	**0.989**	0.974	0.930	0.919
EthanolConcentration	0.293	0.373	0.323	0.323	0.304	0.236	0.430	0.285	0.297	0.312	0.308	**0.450**	0.430	0.337	0.323
FaceDetection	0.519	0.545	0.529	0.556	-	0.528	0.545	0.544	0.536	0.602	0.501	0.638	0.612	0.625	**0.641**
FingerMovements	0.550	0.580	0.530	0.530	0.520	0.540	0.490	0.460	0.470	0.580	0.480	0.520	0.550	0.590	**0.620**
HandMovementDirection	0.278	0.365	0.231	0.378	0.306	0.270	0.365	0.243	0.324	0.338	0.338	0.486	0.392	**0.675**	0.392
Handwriting	0.200	0.286	0.286	0.357	0.316	0.533	**0.605**	0.498	0.249	0.451	0.515	0.596	0.520	0.359	0.281
Heartbeat	0.619	0.663	0.717	0.751	0.658	0.737	0.727	0.751	0.746	0.756	0.683	0.741	0.771	**0.782**	0.780
InsectWingbeat	0.128	0.167	-	0.208	-	0.160	-	0.264	0.469	0.250	0.466	0.179	0.229	**0.687**	0.137
JapaneseVowels	0.924	0.976	0.949	0.965	0.959	0.989	0.973	0.930	0.978	0.984	0.984	0.978	0.986	**0.995**	0.992
Libras	0.833	0.856	0.870	0.850	0.894	0.867	0.878	0.822	0.817	0.856	0.867	0.906	0.922	0.861	**1.000**
LSST	0.456	0.373	0.551	0.568	0.575	0.558	0.590	0.474	0.595	0.590	0.537	0.635	0.653	0.576	**0.976**
MotorImagery	0.510	0.510	0.500	0.590	-	0.540	0.500	0.610	0.500	0.610	0.510	0.460	0.610	0.610	**0.630**
NATOPS	0.850	0.889	0.883	0.939	0.850	**0.944**	0.870	0.822	0.911	0.883	0.928	0.872	0.933	0.939	0.911
PEMS-SF	0.705	0.699	0.711	0.751	0.734	0.688	-	0.734	0.699	0.751	0.682	0.832	0.809	0.930	**0.936**
PenDigits	0.973	0.978	0.977	0.980	0.939	0.983	0.948	0.974	0.979	0.977	**0.989**	0.981	0.967	0.981	0.976
Phoneme	0.104	0.110	0.151	0.175	0.151	0.246	0.190	0.252	0.207	**0.298**	0.233	0.273	0.291	0.111	0.165
RacketSports	0.868	0.803	0.803	0.868	0.842	0.862	0.934	0.816	0.776	0.882	0.855	0.901	0.868	0.796	**0.987**
SelfRegulationSCP1	0.771	0.874	0.775	0.652	0.765	0.846	0.710	0.823	0.799	0.782	0.812	0.867	0.915	**0.961**	0.816
SelfRegulationSCP2	0.483	0.472	0.539	0.550	0.533	0.556	0.460	0.533	0.550	0.578	0.578	0.555	0.506	0.604	**0.622**
SpokenArabicDigits	0.967	0.990	0.963	0.983	0.959	0.956	0.982	0.970	0.934	0.975	0.988	0.997	0.963	**0.998**	0.985
StandWalkJump	0.200	0.067	0.200	0.400	0.333	0.400	0.333	0.333	0.400	0.533	0.467	0.467	0.333	**0.600**	0.533
UWaveGestureLibrary	0.881	0.891	0.903	0.894	0.868	0.884	0.916	0.753	0.759	0.906	0.906	**0.931**	0.785	0.913	0.878
PAMAP2	0.718	0.949	0.683	0.865	0.769	0.885	0.928	0.942	0.938	0.948	0.941	0.931	0.962	0.948	**0.974**
OpportunityGestures	0.655	0.768	0.762	0.574	0.715	0.689	0.553	0.791	0.821	0.730	0.771	0.813	0.809	0.732	**0.830**
OpportunityLocomotion	0.845	0.900	0.859	0.850	0.868	0.859	0.634	0.881	0.874	0.874	0.842	0.875	0.886	0.907	**0.908**
Occupancy [15]	0.496	0.873	0.517	0.844	0.526	0.817	0.556	0.865	0.828	0.852	0.876	0.832	0.878	0.881	**0.883**
Total best accuracy	0	0	1	2	1	2	5	1	0	2	1	6	4	7	**17**
Average accuracy	0.596	0.651	0.658	0.672	0.675	0.686	0.688	0.692	0.693	0.717	0.722	0.732	0.741	0.745	**0.772**
Ours 1-to-1 Wins	32	26	27	23	31	23	25	28	29	25	24	20	21	20	-
Ours 1-to-1 Draws	0	0	2	2	1	2	3	1	1	2	0	2	4	0	-
Ours 1-to-1 Losses	2	8	5	9	2	9	6	5	4	7	10	12	9	14	-
Mean Rank	12.15	8.79	9.65	7.44	10.44	7.59	7.79	9.03	9.41	5.47	7.18	5.18	4.71	5.74	**4.00**

datasets over TST. The closest competitors of TARNet are TST and Rocket, but TARNet still outperforms them on 20 datasets while losing on 14 and 12, respectively. TARNet ranks 1st (lowest "Mean Rank") on average, having a 0.71-point lower average than the 2nd best MiniRocket. Rocket and ShapeNet ranks 3rd and 4th with a 1.18 and 1.47-point higher average, respectively, than TARNet.

The large number of datasets and baselines used makes it highly unlikely for a single model to outperform all other methods on every dataset. For example, TST had the 2nd best "Total best Accuracy" (7) and "Average Accuracy" (0.745), but it ranks 5th across all models, with a 1.74-point higher average than TARNet. This means that for the datasets where TST under-performs, its performance metrics are significantly below those of other baselines, pushing down its "Mean Rank." However, TARNet performs well across all evaluation metrics. Not only does it have the highest "Total best Accuracy" (17) and "Average Accuracy" (0.772), but it also ranks 1st, meaning that for the datasets where TARNet under-performs, it still generates better performance than most of its baselines, pushing up its "Mean Rank". Moreover, from Table 1, we find that on datasets where TARNet under-performs, the winning methods are in fact different. Considering that no single baseline is consistently better than TARNet, as illustrated by the baselines' low number of best accuracies, low average accuracies and high mean rank, we argue that TARNet is the new benchmark for time-series classification.

Moreover, TARNet achieves the best accuracy across a diverse set of data characteristics. For example, TARNet has the best accuracy for Atrial Fibrillation and Occupancy with 15 and 1.2m+ training data points, respectively, for RacketSports and Cricket with sequence length of 30 and 1197, respectively, for Epilepsy and FaceDetection with 3 and 44 features, respectively and for MotorImagery and OpportunityGestures with 2 and 17 classes, respectively.

4.4 Regression

We compare regression results against all the baselines reported by TST [37]. Table 2 shows the Root Mean Squared Error of the models. TARNet ranks 1st on three and 2nd on two datasets, which

Table 2: Root Mean Squared Error (RMSE) Performance of TARNet and baselines on regression datasets from Monash University, UEA, UCR Time Series Regression Archive [27]. We mark the best and second best values. Baselines are presented in descending order (left to right) by mean rank. Avg.Rel.Diff.Mean: Average Relative Difference from Mean over all models, e.g. -0.3 means that the model on average attains 30% lower RMSE than the average model performance. Higher Total best loss and Ours 1-to-1 Wins is better. Lower Ours 1-to-1 Draws, Ours 1-to-1 Losses, Mean Rank, and Avg.Rel.Diff.Mean is better.

Dataset	1-NN-DTWD	1-NN-ED	5-NN-ED	5-NN-DTWD	SVR	ResNet	FCN	Rocket	Inception	RF	XGB	TST	TARNet
AppliancesEnergy	6.036	5.231	4.227	4.019	3.457	3.065	2.865	2.299	4.435	3.455	3.489	2.375	**2.173**
BenzeneConcentration	4.983	6.535	5.844	4.868	4.790	4.061	4.988	3.360	1.584	0.855	0.637	0.494	**0.481**
BeijingPM10	139.134	139.229	115.502	115.502	110.574	95.489	94.438	120.057	96.749	94.072	93.138	**86.866**	90.482
BeijingPM25	88.256	88.193	74.156	72.717	75.734	64.462	59.726	62.769	62.227	63.301	59.495	**53.492**	60.271
LiveFuelMoisture	57.111	58.238	46.331	46.290	43.021	51.632	47.877	41.829	51.539	44.657	44.295	43.138	**41.091**
IEEEPPG	37.140	33.208	27.111	33.572	36.301	33.150	34.325	36.515	**23.903**	32.109	31.487	27.806	26.372
Total best loss	0	0	0	0	0	0	0	0	1	0	0	2	**3**
Ours 1-to-1 Wins	6	6	6	6	6	6	5	6	5	6	5	4	-
Ours 1-to-1 Draws	0	0	0	0	0	0	0	0	0	0	0	0	-
Ours 1-to-1 Losses	0	0	0	0	0	0	1	0	1	0	1	2	-
Mean Rank	12.167	11.833	8.833	8.833	8.000	7.333	7.000	6.500	6.500	5.500	4.333	2.500	**1.833**
Avg.Rel.Diff.Mean	0.355	0.379	0.153	0.125	0.097	0.006	0.022	-0.047	-0.107	-0.171	-0.196	-0.302	**-0.313**

Table 3: Ablation study of TARNet

	TARNet-Random	TARNet-Top μ	TARNet
	Results on 34 classification datasets		
Total best accuracy	6	9	**31**
Average accuracy	0.752	0.741	**0.772**
Ours 1-to-1 Wins	28	25	-
Ours 1-to-1 Draws	5	7	-
Ours 1-to-1 Losses	1	2	-
Mean Rank	2.206	2.176	**1.088**
	Results on 6 regression datasets		
Total best loss	0	1	**5**
Ours 1-to-1 Wins	6	5	-
Ours 1-to-1 Draws	0	0	-
Ours 1-to-1 Losses	0	1	-
Mean Rank	2.667	2.167	**1.167**
Avg.Rel.Diff.Mean	0.046	0.014	**-0.060**

is better than what any of the baseline models achieve. For the overall rank, TARNet achieves an average rank of 1.833, setting it clearly apart from all other models; the overall second best model, TST [37] has an average rank of 2.5; XGB, Inception, and FCN (which outperformed TARNet on one dataset) on average ranks 4.333, 6.5, and 7, respectively. Both TST [37] and TARNet use a similar transformer backbone model which explains the small difference in Avg.Rel.Diff.Mean scores. However, TARNet still outperforms TST and all other baseline models by attaining 31.3% lower RMSE on average than the mean RMSE among all models. Considering that TARNet achieves the highest number of best losses, lowest mean rank, and lowest Avg.Rel.Diff.Mean in Table 2, we argue that TARNet is the new benchmark for time-series regression.

Although TST [37] pretrains and finetunes on the same dataset, the data reconstruction and the supervised end-task runs *sequentially*, slowing down training time. However, TARNet trains both tasks, T_{TAR} and T_{END} *parallely*. Hence, not only TARNet outperforms TST on the end-task but it also trains faster than TST.

4.5 Ablation Study

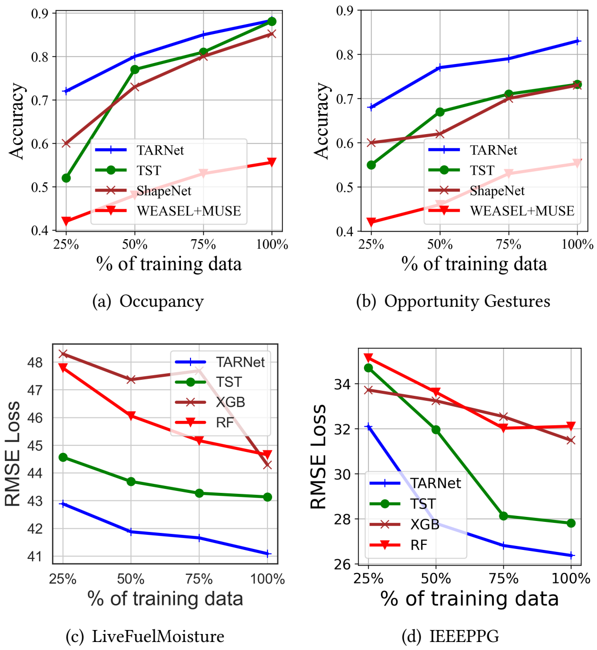
We justify our design choices of M through ablation study results on classification and regression tasks in Table 3. TARNet-Random uses the same architecture as TARNet but instead masks timestamps randomly and reconstructs them, giving substandard performance. TARNet-Top μ selects timestamps corresponding to the top $\lfloor \mu S \rfloor$ values in σ and masks them from X for reconstruction. This does not lead to a clear improvement which may be attributed to *overfitting*, as explained in Section 3.5. This prompts sampling to TARNet-Top μ while selecting the timestamps to mask from the set of important timestamps, resulting in TARNet. To ensure a fair comparison, we maintain the same set of hyper-parameters across all ablation models for each dataset. Table 3 shows that TARNet has the highest average accuracy, most number of datasets with highest accuracy and lowest loss, and lowest mean rank. TARNet combines ideas from both TARNet-Random and TARNet-Top μ to counter their individual drawbacks and yields better performance.

4.6 Can T_{TAR} compensate for limited labeled training data?

We study whether under data-deficient environments TARNet can make better use of limited data compared to baselines. This will illustrate if the knowledge gained during reconstruction, T_{TAR}, can compensate for a lack of labeled data to train the end task, T_{END}.

We choose occupancy and human gestures datasets for classification. As Figure 2 (a) and (b) show, the accuracy of all models increases as the amount of training data increases. Particularly, TARNet has a steep rise for both datasets, signifying that the greatest improvement occurs with low quantity of training data. Similarly, we choose LiveFuelMoisture and IEEEPPG datasets for regression. As Figure 2 (c) and (d) show, the RMSE Loss of all models decreases as the amount of training data increases. Even with just 25% training data, TARNet achieves significantly lower loss than any baselines. It achieves superior performance over all baselines at all quantities of training data, for both classification and regression.

Both TST and TARNet can leverage additional information learnt though reconstruction to compensate for the lack of labeled data, resulting in better performance over other baselines. However, making the reconstruction task-aware improves the performance of TARNet over TST. For example, in Occupancy, TARNet achieves the same performance with 50% training data, which TST and ShapeNet require 75% training data to achieve. Similarly, for LiveFuelMoisture



(a) Occupancy (b) Opportunity Gestures

(c) LiveFuelMoisture (d) IEEEPPG

Figure 2: (a) and (b) show classification accuracy, and (c) and (d) show regression RMSE Loss against % of training data.

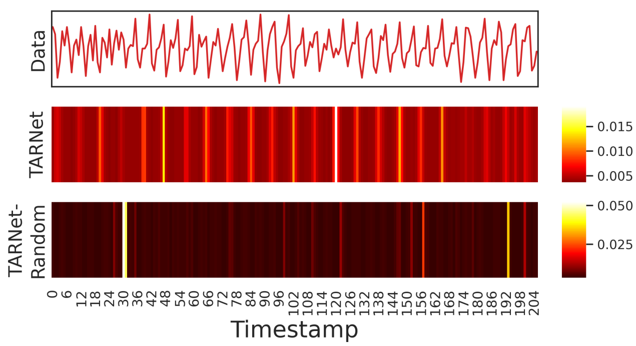


Figure 3: σ plotted as heatmap for Epilepsy.

and IEEEPPG, TARNet achieves lower RMSE with just 25% and 50% training data, respectively, than TST does with 100% training data.

4.7 Explaining Masking Strategy, M

We provide two real-world case studies to show why a task-aware reconstruction learnt through a data-driven masking strategy, M, is superior to a reconstruction learnt through random masking. For qualitative analysis, we show normalized aggregate attention, σ, computed from attention maps of Transformer during T_{END}.

Case Study I: Epilepsy. Figure 3 shows a time-series plot of an accelerometer data from a person conducting the activity of "Sawing" (classification label). Following the time-series plot are the σ scores, as discovered by TARNet and TARNet-Random. Sawing involves strong periodic motion of the hand as illustrated by the time-series plot. Figure 3 shows that a random-masking based auto-regressive task (TARNet-Random) could not capture this inherent periodicity in the data, which TARNet could successfully decipher. Therefore, the accuracy achieved by TARNet and TARNet-Random is 1 and 0.75, respectively. Being able to selectively mask "important" timestamps during reconstruction in a data-driven manner enables TARNet to effectively capture the domain-specific properties from the data, leading to better classification performance.

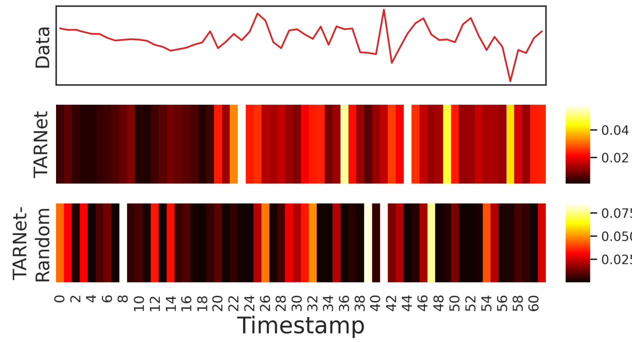


Figure 4: σ plotted as heatmap for Face Detection.

Case Study II: Face Detection. A person is shown a face image or a scrambled image and her MEG readings are recorded. The task is to determine what the person saw (classification) based on the collected MEG data. The MEG recording (response) is collected over 1.5-second but the image (stimulus) is only shown 0.5-seconds after the MEG has started recording. Figure 4 shows the time-series plot of a sample MEG data. Since the entire 1.5-second corresponds to 62 timestamps, this means that no stimulus was provided to the subject for the first 20 timestamps (0.5-seconds). So the discriminatory MEG response, important for classification, is received from 20-th timestamp onward, as illustrated by the onset of sudden fluctuation in signal strength. Figure 4 shows that TARNet assigns high σ values around the 20-th timestamp and can clearly infer the signal arrival time from the MEG response. TARNet discriminates between the "unimportant" and "important" timestamps for classification by assigning higher average attention per timestamp for times greater than 20 than to those before 20. However, TARNet-Random fails to infer such task-specific domain properties from the data and assigns attention weights randomly across time. Hence, TARNet-Random achieves an accuracy of 0.607, whereas TARNet achieves 0.641.

The two case studies substantiate why using M to decide which timestamps to mask during reconstruction is important. Representations learnt through reconstructing "important" timestamps reflect some domain-specific inherent properties in the data, as illustrated by how the attention scores have been assigned. Such domain properties are relevant to the end task and can clearly lead to performance improvement on the end task, as illustrated in Table 1 and 2. We also highlight that the utility of self-attention goes beyond computing internal data representation of a model to improve performance [29] or providing meaningful explanations [17, 34]. In addition, self-attention can also be used to integrate simple and intuitive data-driven techniques into deep learning frameworks.

5 DISCUSSION AND CONCLUSIONS

We have proposed a task-aware reconstruction technique to improve end-task performance for a time series. In particular, we use attention score distribution to identify timestamps important to an end task. We then sample from those important timestamps and mask them from the data for reconstruction, making the reconstruction *end task-aware*. These tasks are trained alternately, sharing parameters in the same model, thereby enabling the representation learnt through reconstruction to improve end-task performance. Experimental results show that TARNet outperforms the state-of-the-art baselines for both classification and regression tasks. The ablation study highlights the essence of our design choices for the data masking technique, and the case study observations show how TARNet captures the intrinsic task-specific properties of data.

Additional unlabeled data can help to improve TARNet. Although the data reconstruction task is fully unsupervised, it is driven by the end task that requires labeled data. In the future, we wish to explore such task-aware representations under data shift problem and in the presence of outliers.

ACKNOWLEDGMENTS

This work was supported in part by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, under award 2018-JU-2779. This work was also sponsored in part by National Science Foundation Convergence Accelerator under award OIA-2040727 as well as generous gifts from Google, Adobe, and Teradata. Ranak is partially funded by a Graduate Prize Fellowship from Halıcıoğlu Data Science Institute. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and should not be interpreted as necessarily representing the views, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes not withstanding any copyright annotation hereon.

REFERENCES

[1] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. 2018. The UEA multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075* (2018).
[2] Yue Bai, Lichen Wang, Zhiqiang Tao, Sheng Li, and Yun Fu. 2021. Correlative Channel-Aware Fusion for Multi-View Time Series Classification. In *AAAI*.
[3] Mustafa Gokce Baydogan and George Runger. 2015. Learning a symbolic representation for multivariate time series classification. *Data Mining and Knowledge Discovery* 29, 2 (2015), 400–422.
[4] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *KDD*. 785–794.
[5] Angus Dempster, François Petitjean, and Geoffrey I Webb. 2020. ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery* 34, 5 (2020), 1454–1495.
[6] Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. 2021. MINIROCKET: A very fast (almost) deterministic transform for time series classification. In *KDD*.
[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[8] Abhilash Dorle, Fangyu Li, Wenzhan Song, and Sheng Li. 2020. Learning Discriminative Virtual Sequences for Time Series Classification. In *CIKM*. 2001–2004.
[9] Harris Drucker, Chris JC Burges, Linda Kaufman, Alex Smola, Vladimir Vapnik, et al. 1997. Support vector regression machines. *NIPS* 9 (1997), 155–161.
[10] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
[11] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. 2021. Time-Series Representation Learning via Temporal and Contextual Contrasting. *arXiv preprint arXiv:2106.14112* (2021).
[12] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *Data mining and knowledge discovery* 33, 4 (2019), 917–963.
[13] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. 2020. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery* 34, 6 (2020).
[14] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. 2019. Unsupervised scalable representation learning for multivariate time series. *arXiv preprint arXiv:1901.10738* (2019).
[15] Dezhi Hong, Jorge Ortiz, Kamin Whitehouse, and David Culler. 2013. Towards automatic spatial verification of sensor placement in buildings. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*. 1–8.
[16] Lu Hou, James Kwok, and Jacek Zurada. 2016. Efficient learning of timeseries shapelets. In *AAAI*, Vol. 30.
[17] Tsung-Yu Hsieh, Suhang Wang, Yiwei Sun, and Vasant Honavar. 2021. Explainable Multivariate Time Series Classification: A Deep Neural Network Which Learns to Attend to Important Variables As Well As Time Intervals. In *WSDM*.
[18] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. 2019. Multivariate LSTM-FCNs for time series classification. *Neural Networks* 116 (2019), 237–245.
[19] Dongha Lee, Seonghyeon Lee, and Hwanjo Yu. 2021. Learnable Dynamic Temporal Pooling for Time Series Classification. *arXiv preprint arXiv:2104.02577* (2021).
[20] Guozhong Li, Byron Choi, Jianliang Xu, Sourav S Bhowmick, Kwok-Pan Chun, and Grace LH Wong. 2021. Shapenet: A shapelet-neural network approach for multivariate time series classification. In *AAAI*, Vol. 35. 8375–8383.
[21] Haoran Liang, Lei Song, Jianxing Wang, Lili Guo, Xuzhi Li, and Ji Liang. 2021. Robust unsupervised anomaly detection via multi-time scale DCGANs with forgetting mechanism for industrial multivariate time series. *Neurocomputing* 423 (2021), 444–462.
[22] Qianli Ma, Zhenjing Zheng, Jiawei Zheng, Sen Li, Wanqing Zhuang, and Garrison W Cottrell. 2021. Joint-Label Learning by Dual Augmentation for Time Series Classification. In *AAAI*, Vol. 35. 8847–8855.
[23] Qianli Ma, Wanqing Zhuang, Sen Li, Desen Huang, and Garrison Cottrell. 2020. Adversarial dynamic shapelet networks. In *AAAI*, Vol. 34. 5069–5076.
[24] Patrick Schäfer and Ulf Leser. 2017. Multivariate time series classification with WEASEL+ MUSE. *arXiv preprint arXiv:1711.11343* (2017).
[25] Mohammad Shokoohi-Yekta, Jun Wang, and Eamonn Keogh. 2015. On the non-trivial generalization of dynamic time warping to the multi-dimensional case. In *ICDM*. SIAM, 289–297.
[26] Vladimir Svetnik, Andy Liaw, Christopher Tong, J Christopher Culberson, Robert P Sheridan, and Bradley P Feuston. 2003. Random forest: a classification and regression tool for compound classification and QSAR modeling. *Journal of chemical information and computer sciences* 43, 6 (2003), 1947–1958.
[27] Chang Wei Tan, Christoph Bergmeir, Francois Petitjean, and Geoffrey I Webb. 2020. Monash university, uea, ucr time series regression archive. *arXiv e-prints* (2020), arXiv–2006.
[28] Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. 2021. Unsupervised representation learning for time series with temporal neighborhood coding. *arXiv preprint arXiv:2106.00750* (2021).
[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.
[30] Zhiguang Wang, Weizhong Yan, and Tim Oates. 2017. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*. IEEE, 1578–1585.
[31] Xiaopeng Xi, Eamonn Keogh, Christian Shelton, Li Wei, and Chotirat Ann Ratanamahatana. 2006. Fast time series classification using numerosity reduction. In *ICML*. 1033–1040.
[32] Chao-Han Huck Yang, Yun-Yun Tsai, and Pin-Yu Chen. 2021. Voice2series: Reprogramming acoustic models for time series classification. In *ICML*.
[33] Lexiang Ye and Eamonn Keogh. 2009. Time series shapelets: a new primitive for data mining. In *KDD*. 947–956.
[34] Ye Yuan, Guangxu Xun, Fenglong Ma, Yaqing Wang, Nan Du, Kebin Jia, Lu Su, and Aidong Zhang. 2018. Muvan: A multi-view attention network for multivariate temporal data. In *ICDM*. IEEE, 717–726.
[35] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. 2021. TS2Vec: Towards Universal Representation of Time Series. *arXiv preprint arXiv:2106.10466* (2021).
[36] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, and Bixiong Xu. 2021. Learning Timestamp-Level Representations for Time Series with Hierarchical Contrastive Loss. *arXiv preprint arXiv:2106.10466* (2021).
[37] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. [n. d.]. A Transformer-based Framework for Multivariate Time Series Representation Learning. In *KDD, pages=2114–2124, year=2021*.
[38] Xuchao Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. 2020. Tapnet: Multivariate time series classification with attentional prototypical network. In *AAAI*.
[39] Yi Zheng, Qi Liu, Enhong Chen, Yong Ge, and J Leon Zhao. 2014. Time series classification using multi-channels deep convolutional neural networks. In *International conference on web-age information management*. Springer, 298–310.