Generating Small Anomaly Detection Models through Distillation of Long-Term Dependency

Seonyoung Kim
School of Computing, Korea Advanced Institute of Science and Technology
sykim@dbserver.kaist.ac.kr

Abstract Time-series anomaly detection is the task of identifying data points that exhibit different characteristics from normal data. Existing attention-based models require a large number of parameters, and their memory and time complexity scales with the input sequence length, making them impractical for resource-constrained environments. Common solutions involve reducing the model size or using a sliding window, but these often degrade performance by diminishing model capacity or failing to capture long-term dependencies. In this paper, we propose a method that enables a lightweight attention-based model to learn long-term dependencies effectively even from short input sequences. Our approach leverages knowledge distillation and introduces a novel mechanism called the Long-Term Accumulator (LTA). Experimental results demonstrate that our approach achieves over 95% parameter reduction, more than 50% shorter input sequences, and a 50% decrease in inference time, all while preserving the model's F1 score with negligible performance loss.

Key words: Anomaly detection, Time-series data, Short data, Network compression, Long-term dependency, Knowledge distillation

1. Introduction

With the proliferation of IoT sensors and real-time monitoring systems in modern industrial environments, massive amounts of time-series data are now being continuously collected. During system operation, abnormal conditions may arise due to internal faults or external intrusions, and the task of identifying such anomalous data is referred to as anomaly detection [1, 2]. Failure to detect anomalies in a timely manner can lead to system errors or shutdowns, potentially causing significant economic loss, underscoring the critical importance of anomaly detection techniques.

The rapid advancement of deep learning [3, 4, 5] has catalyzed the development of numerous sophisticated techniques for anomaly detection. Nevertheless, anomaly detection in time-series data remains inherently challenging for several reasons. First, in real-world industrial settings, time-series data are rarely labeled, making unsupervised learning approaches more common than supervised ones. Second, time-series data can be exceptionally long, requiring models to capture long-term dependencies [6]. However, as network depth increases, gradients are prone to vanishing or exploding, which can prevent information from being effectively propagated through the model and thus impede proper training.

Recently, autoencoder-based models [7, 8, 9, 10] have been widely adopted for time-series anomaly detection. In particular, the advent of attention-based autoencoders has enabled the use of more complex unsupervised models. However, the standard attention mechanism suffers from two fundamental limitations. First, computing attention across the temporal dimension requires a large number of parameters. Second, the memory and computational time scale quadratically with the input sequence length, both in the attention mechanism itself and in data processing. Consequently, such models are often impractical for deployment in resource-constrained environments where insufficient memory or high inference latency can prevent timely anomaly detection.

To mitigate these issues, one potential solution is to reduce the model size, though this often leads to performance degradation due to diminished model capacity. Another common approach is to employ a sliding-window technique to shorten the input sequence length. However, because the model can only learn contextual information within each window, an inadequate window length makes it difficult to capture long-term dependencies, again resulting in a loss of performance.

In this paper, we propose a method that enables a lightweight, few-layer attention-based model to effectively learn long-term dependencies even when using short input sequences. This approach simultaneously reduces resource consumption while enhancing detection performance. Our proposed method is based on knowledge distillation and introduces a novel auxiliary component, the **L**ong-**T**erm **A**ccumulator (LTA), designed to retain and propagate critical temporal information.

2. Background

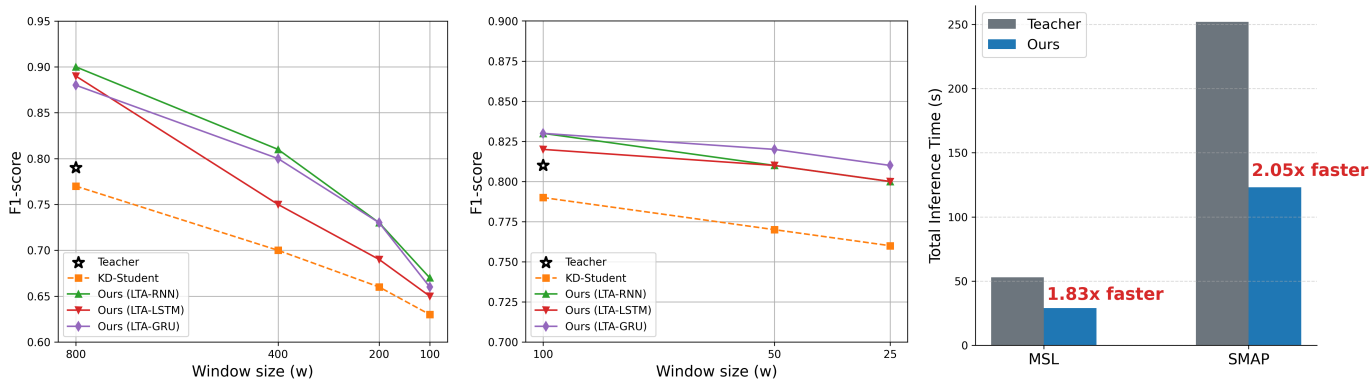
2.1 Time-series Anomaly Detection Methods

Due to the scarcity of labeled time-series data, unsupervised learning methods are widely adopted for anomaly detection. For instance, the work in [7] utilizes a Long Short-Term Memory (LSTM) network to predict future values of multivariate time-series data, identifying anomalies based on the deviation between the predicted and actual values. An LSTM-based autoencoder has been used for anomaly detection, where both the encoder and decoder are composed of LSTM layers [8]. The model is trained exclusively on normal data; the encoder compresses the input into a lower-dimensional latent vector, from which the decoder reconstructs the original data. Anomalies are then detected by measuring the reconstruction error between the original and reconstructed data. Similarly, a network using Gated Recurrent Units (GRUs) and a Variational Autoencoder (VAE) has been constructed for anomaly detection [9]. The model learns the temporal dependencies of normal data, and anomalies are identified based on the reconstruction probability.

Our work utilizes the Multi-Scale Convolutional Recurrent Encoder-Decoder (MSCRED) model [10]. MSCRED performs anomaly detection by leveraging both the temporal dependencies in time-series data and the inter-variable correlations. It also provides an interpretation of anomaly severity by analyzing the cause of the anomaly. The overall architecture is illustrated in Figure 2.

To capture the inter-variable correlations, the model first con-

This paper is an abridged English version of the author's Master's Thesis, originally written in Korean and submitted to KAIST in 2022.



(a) MSL dataset (b) SMAP dataset (c) Inference time comparison

Figure 1: Performance and inference efficiency of the proposed method on the MSL and SMAP datasets. The LTA-based models consistently outperform the KD-Student baseline, achieving accuracy close to the teacher model while also reducing inference time by up to 2.05×.

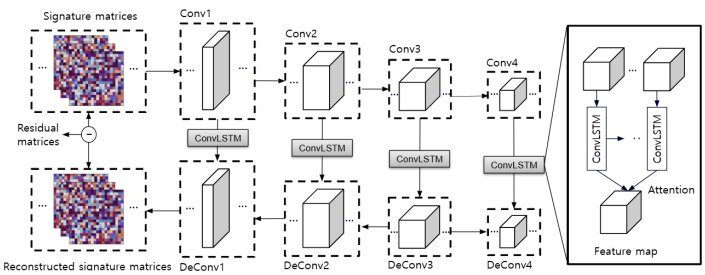


Figure 2: Architecture of MSCRED.

structs a signature matrix, denoted as M_t, by calculating the inner product between variables within a sliding window. The element $m_t^{(i,j)}$ at the i-th row and j-th column of M_t represents the correlation between the i-th and j-th variables, which can be expressed as:

$$m_t^{(i,j)} = \frac{X_{t-\ell:t}^i \cdot \left(X_{t-\ell:t}^j\right)^T}{\ell}, \tag{1}$$

where ℓ is the length of the window, and $X_{t-\ell:t}^i = \{x_{t-\ell}^i, x_{t-\ell+1}^i, ..., x_t^i\}$ represents the window for the i-th variable at time t. Notably, MSCRED constructs these signature matrices using three sliding windows of different lengths.

The encoder, composed of four convolutional layers, processes these signature matrices to generate feature maps. These feature maps then serve as input to an attention-based Convolutional LSTM (ConvLSTM). The attention mechanism applied in the model assigns higher weights to feature maps from previous time steps that are more relevant to the current feature map and lower weights to those that are less relevant. This can be expressed as:

$$\hat{\mathcal{F}}_{(t,n)} = \sum_{i=t-k}^{t} \frac{\left(\mathcal{F}_{(t,n)}\right)^T \mathcal{F}_{(i,n)}}{\mathcal{C}} \cdot \mathcal{F}_{(i,n)}, \tag{2}$$

$$\mathcal{C} = \sum_{j=t-k}^{t} \exp\left(\frac{\left(\mathcal{F}_{(t,n)}\right)^T \mathcal{F}_{(j,n)}}{\beta}\right), \tag{3}$$

where $\mathcal{F}_{(i,n)}$ is the feature map produced by the n-th layer of the ConvLSTM at time step t, and $\mathcal{F}_{(t,n)}$ is the reconstructed feature map after applying the attention mechanism. β is a rescale factor, and k is the number of feature maps from previous time steps used as input to the ConvLSTM. This reconstructed feature map is then fed into the decoder.

The decoder, consisting of four deconvolutional layers, reconstructs the signature matrix. The loss function is defined as the residual matrix, which is the difference between the original signature matrix and the final reconstructed signature matrix from the decoder.

For anomaly detection, the residual matrix Υ_t is utilized. This is formulated as follows:

$$\Upsilon_t = S_t^1 - \hat{S}_t^1, \tag{4}$$

where S_t^i represents the i-th original signature matrix (out of three scales) at time t, and $\hat{S}_t^i$ is the corresponding i-th reconstructed signature matrix from the decoder. If the residual matrix Υ_t contains more than n elements exceeding a predefined threshold τ, the corresponding matrix is detected as an anomaly.

2.2 Knowledge Distillation

Knowledge Distillation (KD) is a network compression technique [11]. The primary objective of KD is to enhance the performance of a compact model by transferring the knowledge from a larger, more powerful model. This makes the resulting compact model suitable for deployment in resource-constrained environments, such as mobile applications and real-time systems.

In this paradigm, the large model is referred to as the teacher model, and the compact model is the student model. The teacher's knowledge can be expressed as the softened probability distribution obtained from its softmax layer. This is achieved by applying a temperature hyperparameter, T, to the logits ($\mathbf{n}_t$) before the softmax operation, formulated as $\sigma(\mathbf{n}_t/T)$. The pre-trained teacher model encapsulates knowledge about the training data, which is then distilled into the student. The student model is trained using a composite loss function that combines the standard loss with the ground-truth labels and a distillation loss that aligns the student's

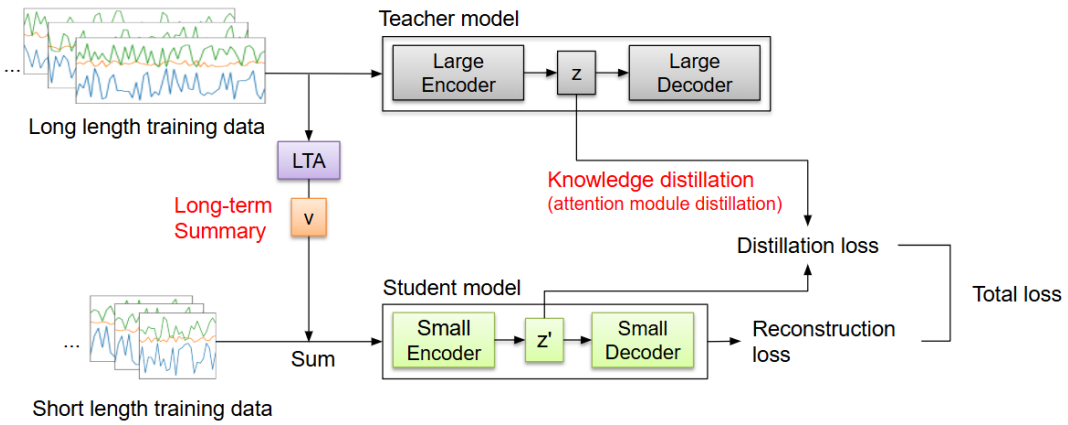


Figure 3: Training architecture of the proposed method.

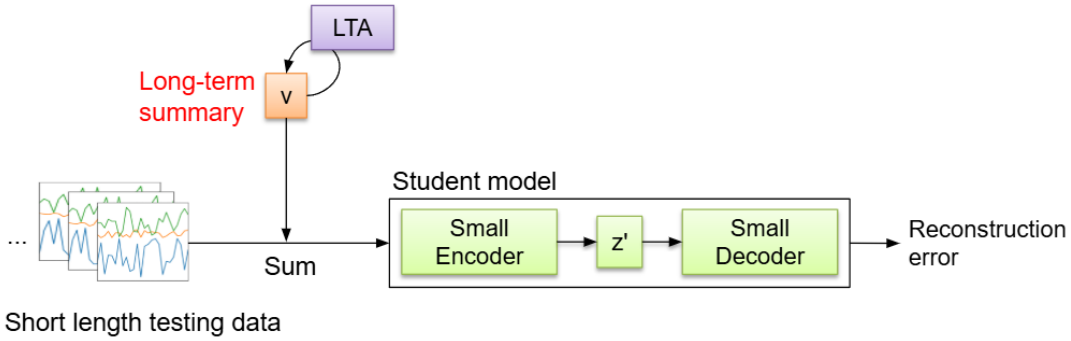


Figure 4: Inference architecture of the proposed model. The input sequence is processed by the encoder stack and the LTA module, then the student model produces outputs without access to ground-truth labels (i.e., no training-time supervision).

softened outputs with those of the teacher.

To address the challenge that training becomes difficult in very deep, non-linear models, [12] proposed a method that transfers knowledge not only from the teacher's final output but also from its intermediate representations. This provides additional hints to guide the student's training process.

Instead of using KD for model compression, [13] employed it for performance enhancement by using teacher and student models of the same size. In this approach, often known as self-distillation, a trained student model becomes the new teacher for a re-initialized student in an iterative retraining process.

The application of KD has also been explored in the time-series domain. For instance, [14] applied knowledge distillation for long-term time-series prediction, specifically to address the cold-start problem that arises from insufficient training data.

3. Proposed Method

3.1 Overall Architecture

Figure 3 illustrates the overall architecture of our proposed model. The teacher model, depicted in the upper part of the figure, receives a long input sequence. The extended length of this input data enables the teacher to effectively learn the long-term dependencies inherent within the time-series.

In contrast, the student model, shown in the lower part of the figure, is a more compact network that processes significantly shorter input sequences compared to the teacher. In our implementation, the teacher's encoder and decoder are each constructed with four layers, whereas the student's encoder and decoder consist of only two layers. To transfer the data-centric knowledge and the understanding of long-term dependencies acquired by the teacher during its training, we employ two key mechanisms.

The first mechanism is the **L**ong-**T**erm **A**ccumulator (LTA). The LTA is a component that is continuously trained on the incoming data stream to generate a **L**ong-**T**erm **S**ummary (LTS). This LTS is a vector that encapsulates information about the entire data history and serves as a hint to the student model, providing it with global context. The student model incorporates this hint by adding the LTS vector to its own short input sequence. This process allows the student to learn long-term dependencies despite its limited direct view of the data.

The second mechanism is the transfer of the intermediate feature map (denoted as z in Figure 3), generated within the teacher's attention module, via knowledge distillation. The teacher model, with its greater number of parameters and access to longer data sequences, possesses a more comprehensive knowledge of the data and exhibits superior performance. By distilling this rich representation, we enable the student model to learn the nuanced patterns captured by the teacher.

The fully trained student model is deployed in conjunction with the LTA, as illustrated in the testing process in Figure 4. As new, unseen test data arrives, it is first fed into the LTA to generate the corresponding LTS vector. This vector is then added to the test

data sequence, just as in the training phase. The student model takes this combined input to produce its final output, which is then used to detect anomalous data points.

3.2 Long-Term Accumulator (LTA)

The **L**ong-**T**erm **A**ccumulator (LTA) is an encoder designed to learn long-term dependency information from the entire dataset and generate a vector to transfer this knowledge to the student model. The vector generated by this component is termed the **L**ong-**T**erm **S**ummary ('LTS'). The LTA is trained concurrently with the student model.

To learn from the complete data history, the LTA processes the same long input windows as the teacher model, accumulating information sequentially. To maintain temporal continuity, the final hidden state after processing one window is used as the initial hidden state for the subsequent window. This process can be formulated as follows:

$$h_t = LTA(h_{t-1}, x_t), \tag{5}$$

$$(h_t, c_t) = LTA((h_{t-1}, c_{t-1}), x_t), \tag{6}$$

where x_t represents the data point at time t, h_t is the hidden state, and c_{t-1} is the cell state. For models that include an internal cell state, such as LSTMs, the update is described by Equation (4), while simpler recurrent structures like RNNs use Equation (3). Given that the LTA must capture temporal information, it is implemented as a sequence model; suitable architectures include RNNs, LSTMs, or GRUs.

In our proposed method, the LTS is incorporated by adding it to the student's input. Specifically, the MSCRED-based student model first constructs its signature matrix, M_t, from its short input window. Concurrently, the LTS vector is used to generate a corresponding signature matrix, which we denote as β_t. The final input to the student model is the element-wise sum of these two matrices. The resulting input signature matrix for the student model, S_t, is formulated as follows:

$$S_t = M_t + \beta_t. \tag{7}$$

The LTS is also utilized during the inference phase. A signature matrix is constructed from the incoming test data. Simultaneously, the LTA processes the same test data to generate an LTS, which is then converted into its own signature matrix. These two matrices are summed to form the final input for the student model. Anomaly detection is then performed based on the reconstructed signature matrix produced by the model.

3.3 Knowledge Distillation

We apply a knowledge distillation technique to transfer the rich data representations and long-term dependency information captured by the teacher model to the student model. Specifically, we distill the intermediate feature map generated by the teacher model's attention mechanism. As the teacher is trained on significantly longer sequences than the student, its feature maps are expected to contain more comprehensive knowledge regarding long-term dependencies. The feature map produced after the attention mechanism, in particular, serves as a dense representation of this knowledge, enabling the student to learn these complex patterns by proxy.

In our architecture, the teacher model is composed of four attention-based ConvLSTM layers (ConvLSTM1 through ConvLSTM4). We select the feature map generated after the second layer, ConvLSTM2, to be the source of the distilled knowledge. Correspondingly, the student model consists of two attention-based ConvLSTM layers. The student learns by aligning the output feature map from its own ConvLSTM2 with the feature map transferred from the teacher.

This is achieved by incorporating a distillation loss term into the student model's overall loss function. This term minimizes the discrepancy between the student's feature map and the teacher's, thereby compelling the student to learn the knowledge and long-term dependency information encapsulated within the teacher's richer representation.

3.4 Loss Function

The loss function for the student model is composed of two components: one for comparing the signature matrices and another for comparing the feature maps. This is formulated as follows:

$$\mathcal{L} = \sum_t (1-\lambda)\, \mathcal{L}_{MSE}(S_t, \hat{S}_t) + \lambda\, \mathcal{L}_{NormMSE}\left(\hat{\mathcal{F}}_s^{(t,1)}, \hat{\mathcal{F}}_t^{(t,1)}\right), \tag{8}$$

where $\mathcal{L}_{MSE}$ is the mean squared error function and $\mathcal{L}_{NormMSE}$ denotes the mean squared error between L2-normalized feature maps. S_t denotes the student model's input signature matrix, and $\hat{S}_t$ is the signature matrix reconstructed by the model. $\hat{\mathcal{F}}_s^{(t,1)}$ is the feature map from the student model, while $\hat{\mathcal{F}}_t^{(t,1)}$ is the feature map from the teacher model. λ is a hyperparameter.

4. Experiment

4.1 Experimental environment

4.1.1 Data

In this study, we conducted experiments using two public datasets: Mars Science Laboratory (MSL) rover and Soil Moisture Active Passive (SMAP) [15]. Both datasets consist of real-world telemetry data provided by NASA, originating from actual spacecraft. The data is divided into training and test sets. For both datasets, anomalous data are present only within the test sets, which are fully labeled. The detailed statistics of the datasets are summarized in Table 2.

4.1.2 Evaluation Metrics

The evaluation of our experiments is based on precision, recall, and the F1 score. We focused our experiments on maximizing the F1 score, as it provides a balanced measure that considers both precision and recall.

To evaluate detection performance, we adopted the point-adjust approach [16]. In this methodology, if an algorithm successfully detects even a single point within a continuous sequence of anomalies (an anomaly segment), the entire segment is considered to have been correctly identified. This approach is highly relevant for real-world industrial scenarios where anomalies often occur consecutively, and it is more practical than requiring the detection of every single anomalous point.

Table 1: Model performance on MSL and SMAP datasets

(a) MSL

Method		w = 800			w = 400			w = 200			w = 100		
		Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
Base		0.75	0.85	0.79	–	–	–	–	–	–	–	–	–
$\text{Base}_{\text{small}}$		0.73	0.82	0.77	0.61	0.81	0.70	0.72	0.61	0.66	0.59	0.69	0.63
Ours	LTA_{RNN}	0.90	0.92	0.90	0.79	0.84	0.81	0.69	0.78	0.73	0.69	0.66	0.67
	LTA_{LSTM}	0.86	0.95	0.89	0.72	0.78	0.75	0.65	0.75	0.69	0.63	0.68	0.65
	LTA_{GRU}	0.86	0.91	0.88	0.75	0.88	0.80	0.76	0.70	0.73	0.63	0.71	0.66

(b) SMAP

Method		w = 100			w = 50			w = 25		
		Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
Base		0.87	0.76	0.81	–	–	–	–	–	–
$\text{Base}_{\text{small}}$		0.80	0.78	0.79	0.70	0.86	0.77	0.67	0.86	0.76
Ours	LTA_{RNN}	0.89	0.78	0.83	0.85	0.79	0.81	0.82	0.83	0.80
	LTA_{LSTM}	0.87	0.79	0.82	0.81	0.83	0.81	0.75	0.86	0.80
	LTA_{GRU}	0.84	0.81	0.83	0.87	0.77	0.82	0.80	0.84	0.81

Table 2: Statistics of MSL and SMAP datasets. The table shows common properties, training data points, and test set composition including anomaly ratios.

Data		MSL	SMAP
Common	# dimension	55	25
Train	# data points	58,317	135,183
Test	# normal points	65,963	372,921
	# abnormal points	7,766	54,696
	Total test points	73,729	427,617
	Anomaly ratio (%)	10.53	12.79

4.1.3 Experimental Setup

For the MSL dataset, the teacher model was trained with an input sequence length of 800. The student model was subsequently evaluated using four different input lengths: 800, 400, 200, and 100. For the SMAP dataset, the teacher model used an input length of 100, while the student model was trained with three different lengths: 100, 50, and 25.

The models were implemented using PyTorch and trained using the Adam optimizer. The learning rate was set to 0.001 for the teacher model and 0.0002 for the student model. To mitigate experimental error and tune hyperparameters, we utilized a toolkit provided by Microsoft [17]. Using this tool, the hyperparameter λ in the loss function was randomly sampled from the range [0.3, 0.7]. The length of the LTS was set to 10. We conducted experiments for the LTA using three different architectures: RNN, LSTM, and GRU.

4.2 Experimental Results

To evaluate our proposed method, we designed our experiments to answer the following three research questions (RQs):

RQ1) Is the proposed method competitive against larger models that use longer input sequences? RQ2) How efficient is the proposed method in terms of memory and computation time? RQ3) What is the performance impact of each component of the proposed method?

4.2.1 Proposed Method Performance (RQ1)

The results are summarized in Table 1. The experimental procedure was conducted by first training the teacher model and subsequently training the student model. The teacher model is referred to as Base, and the student model baseline is $Base_{small}$. The results for our proposed method are presented as follows: 1) Ours(LTA_{RNN}) denotes the use of an RNN for the LTA; 2) Ours(LTA_{LSTM}) uses an LSTM; and 3) Ours(LTA_{GRU}) uses a GRU.

On the MSL dataset, all variants of our method demonstrated a performance improvement. When comparing Ours(LTA_{RNN}), Ours(LTA_{LSTM}), and Ours(LTA_{GRU}) with the teacher model (Base), the F1 score was over 9% higher. For an input length of 400, the F1 score was approximately 1% higher across all three LTA variants. For lengths of 200 and 100, the F1 scores did not surpass the teacher model. When comparing our method to the $Base_{small}$ model, for an input length of 800, the F1 score increased by over 11%. For an input length of 400, the F1 score increased by over 5%. For a length of 200, the F1 score increased

Table 3: The performance of the components of the model on MSL, SMAP data

(a) MSL

Method	w = 800			w = 400			w = 200			w = 100		
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
$\text{Base}_{\text{small}}$	0.73	0.82	0.77	0.61	0.81	0.70	0.72	0.61	0.66	0.59	0.69	0.63
A.D.	0.78	0.86	0.81	0.64	0.81	0.72	0.68	0.70	0.68	0.60	0.70	0.64
LTA_{RNN}	0.90	0.85	0.87	0.74	0.82	0.78	0.69	0.68	0.68	0.68	0.65	0.66
LTA_{LSTM}	0.84	0.87	0.86	0.70	0.73	0.72	0.67	0.68	0.67	0.60	0.69	0.64
LTA_{GRU}	0.87	0.85	0.85	0.76	0.80	0.78	0.68	0.70	0.69	0.67	0.64	0.65

(b) SMAP

Method	w = 100			w = 50			w = 25		
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
$\text{Base}_{\text{small}}$	0.80	0.78	0.79	0.70	0.86	0.77	0.67	0.86	0.76
A.D.	0.84	0.80	0.81	0.81	0.78	0.79	0.81	0.74	0.77
LTA_{RNN}	0.87	0.77	0.81	0.77	0.82	0.79	0.80	0.75	0.77
LTA_{LSTM}	0.86	0.76	0.80	0.76	0.81	0.78	0.76	0.78	0.77
LTA_{GRU}	0.82	0.81	0.82	0.83	0.80	0.77	0.80	0.80	0.78

by over 3%, and for 100, it increased by over 2%. Within our method's variants, the LTA with RNN achieved the highest F1 score.

On the SMAP dataset, a similar performance improvement was observed. When comparing our method with the teacher model (Base), for an input length of 100, our method's F1 score was over 1% higher, and it also showed superior performance for lengths of 50 and 25. Comparing our method with the $Base_{small}$ model, for an input length of 100, the F1 score increased by over 3%. For lengths of 50 and 25, the F1 score increased by over 4%. Among our variants, the LTA with GRU achieved the highest F1 score.

In summary, these results confirm that our proposed method not only improves upon the baseline student model but also achieves performance that is comparable to or even better than the much larger teacher model.

4.2.2 Efficiency Analysis (RQ2)

This analysis aims to measure the efficiency of our proposed method. To this end, we conducted two experiments: 1) measuring the number of model parameters (Table 5) measuring the inference time (Table 4) on the test data.

Table 4: Time for generating test data

Data	**Method**	**1 data (ms)**	**Total data (s)**	**Ratio**
MSL	Base	37	53	1
	Ours	20	29	0.54
SMAP	Base	29	252	1
	Ours	14	123	0.49

Table 5: The number of parameters of the model

Method	**# parameters**	**Ratio**
Base	6,687,843	1
Ours	408,611	0.061

The teacher model (Base) has approximately 6.7 million parameters, whereas our proposed model (Ours) has only about 400,000 parameters. To express this as a ratio, if we set the teacher model's parameter count to 1, our student model's count is merely 0.061. This shows a significant reduction in model size, which directly correlates with reduced memory usage and computational load during training and inference.

The inference times on the MSL and SMAP test datasets were also measured. We recorded the time to process a single data point and the time to process the entire dataset. For the MSL data, where the Base model uses an input length of 800 and Ours uses a length of 400, our model was faster. For the SMAP data, where Base uses an input of length 100 and Ours uses a length of 50, a similar reduction in inference time was observed. While the difference for a single data point may seem small, this efficiency becomes significant when processing hundreds of thousands of test data points, demonstrating the practical efficiency of our method.

4.2.3 Ablation Study (RQ3)

In this ablation study, we measure the performance impact of the two main components of our model: 1) knowledge distillation (A.D.), and 2) the Long-Term Accumulator (LTA). The results

are summarized in Table 3. The baseline for this experiment is the $Base_{small}$ model. A.D. refers to $Base_{small}$ trained with only attention-based knowledge distillation. LTA_{RNN}, LTA_{LSTM}, and LTA_{GRU} refer to $Base_{small}$ trained with only the respective LTA variant.

For the MSL dataset, applying only knowledge distillation (A.D.) to $Base_{small}$ improved the F1 score by 4%, 2%, 2%, and 1% for input lengths of 800, 400, 200, and 100, respectively. When applying only the LTA variants (LTA_{RNN}, LTA_{LSTM}, LTA_{GRU}), the F1 score for an input length of 800 increased by over 8%; for 400, over 2%; and for 100, over 2%. Overall, the LTA contributed to a greater F1 score improvement.

For the SMAP dataset, applying only knowledge distillation improved the F1 score by approximately 2%, 2%, and 1% for input lengths of 100, 50, and 25, respectively. Applying only the LTA improved the F1 score by approximately 2% for an input length of 100 and over 1% for lengths of 50 and 25.

The fact that applying either knowledge distillation or the LTA individually results in F1 score improvements confirms that each component is effective at enhancing performance.

5. Conclusion and Future Work

In this paper, we proposed a method for training a lightweight attention-based model on short input sequences for time-series anomaly detection by transferring long-term information. To achieve this, we introduced two key techniques: knowledge distillation and a novel component called the Long-Term Accumulator (LTA).

Experiments on two public datasets demonstrated that our method improves the performance of the student model while significantly reducing its resource consumption. Furthermore, we showed that our lightweight model can achieve performance comparable to, and in some cases superior to, a much larger model that consumes significantly more resources. This makes our method well-suited for deployment in resource-constrained environments such as IoT sensors or real-time systems that require low latency.

For future work, it would be valuable to evaluate our method on diverse datasets and explore the application of the LTA to other sequential model architectures.

References

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[2] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[6] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[7] P. Malhotra, L. Vig, G. Shroff, P. Agarwal, *et al.*, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, vol. 89, p. 94, 2015.

[8] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv:1607.00148*, 2016.

[9] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2828–2837, 2019.

[10] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, and N. V. Chawla, "A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 1409–1416, 2019.

[11] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[12] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets. arxiv 2014," *arXiv preprint arXiv:1412.6550*, 2014.

[13] T. Furlanello, Z. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," in *International conference on machine learning*, pp. 1607–1616, PMLR, 2018.

[14] S. Hayashi, A. Tanimoto, and H. Kashima, "Long-term prediction of small time-series data using generalized distillation," in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2019.

[15] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 387–395, 2018.

[16] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 world wide web conference*, pp. 187–196, 2018.

[17] Microsoft, "Neural network intelligence (nni)." `https://github.com/microsoft/nni`, 2018. Accessed: 2021-06-03.