electronics

MDPI

Article

Replacing Batch Normalization with Memory-Based Affine Transformation for Test-Time Adaptation

Jih Pin Yeh [1], Joe-Mei Feng [2,*], Hwei Jen Lin [2,*] and Yoshimasa Tokuyama [3]

[1] National Chung-Shan Institute of Science and Technology, Tao-Yuan County, Taoyuan 325204, Taiwan; yb571107@ms5.hinet.net
[2] Department of Computer Science and Information Engineering, Tamkang University, Taipei 251301, Taiwan
[3] Department of Media and Image Technology, Faculty of Engineering, Tokyo Polytechnic University, Tokyo 164-0012, Japan; tokuyama@mega.t-kougei.ac.jp
* Correspondence: 168984@o365.tku.edu.tw (J.-M.F.); 086204@gms.tku.edu.tw (H.J.L.)

Abstract

Batch normalization (BN) has become a foundational component in modern deep neural networks. However, one of its disadvantages is its reliance on batch statistics that may be unreliable or unavailable during inference, particularly under test-time domain shifts. While batch-statistics-free affine transformation methods alleviate this by learning per-sample scale and shift parameters, most treat samples independently, overlooking temporal or sequential correlations in streaming or episodic test-time settings. We propose LSTM-Affine, a memory-based normalization module that replaces BN with a recurrent parameter generator. By leveraging an LSTM, the module produces channel-wise affine parameters conditioned on both the current input and its historical context, enabling gradual adaptation to evolving feature distributions. Unlike conventional batch-statistics-free designs, LSTM-Affine captures dependencies across consecutive samples, improving stability and convergence in scenarios with gradual distribution shifts. Extensive experiments on few-shot learning and source-free domain adaptation benchmarks demonstrate that LSTM-Affine consistently outperforms BN and prior batch-statistics-free baselines, particularly when adaptation data are scarce or non-stationary.

Keywords: batch normalization; affine transformation; LSTM; test-time adaptation; memory-based learning; domain adaptation; few-shot learning; normalization-free networks; deep neural networks; feature distribution shift

check for updates

Academic Editors: Lanting Fang and Yubo Song

Received: 26 September 2025
Revised: 21 October 2025
Accepted: 22 October 2025
Published: 30 October 2025

Citation: Yeh, J.P.; Feng, J.-M.; Lin, H.J.; Tokuyama, Y. Replacing Batch Normalization with Memory-Based Affine Transformation for Test-Time Adaptation. *Electronics* **2025**, *14*, 4251. https://doi.org/10.3390/electronics14214251

Correction Statement: This article has been republished with a minor change. The change does not affect the scientific content of the article and further details are available within the backmatter of the website version of this article.

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Normalization layers—most notably batch normalization (BN)—have become a cornerstone of modern deep neural networks, stabilizing optimization and improving generalization by controlling the distribution of intermediate features during training [1]. Despite its success, BN implicitly relies on the availability and reliability of batch-level statistics, an assumption that is often violated at test time. In particular, few-shot learning (FSL) and source-free domain adaptation (SFDA) operate under restricted data regimes or distribution shift, where batch composition is small, imbalanced, or even sequential [2–4]. Under such non-i.i.d. conditions, the mismatch between training and testing statistics can degrade performance as running estimates fail to represent new input distributions accurately.

We revisit BN from the perspective of feature-wise affine modulation: beyond computing batch statistics, BN effectively applies a learned scale and shift to features. Prior analysis shows that BN's benefit largely arises from smoothing the optimization landscape

rather than merely mitigating internal covariate shift [5]. This motivates a complementary direction—retain the representational benefits of BN's affine transform while removing explicit dependence on batch moments, and make the modulation adapt over time as test inputs evolve. Concretely, we seek a module that (i) is batch-statistics-free, (ii) adapts per instance without backpropagation at test time, and (iii) can be dropped into standard backbones with minimal changes to the training recipe.

To this end, we propose LSTM-Affine, a memory-driven affine transformation that replaces BN's fixed affine part with parameters generated by a lightweight LSTM conditioned on the temporal context of features [6]. By maintaining hidden states across samples, LSTM-Affine captures slow distribution drift and stabilizes predictions in streaming or episodic evaluations. Unlike approaches that update BN statistics or require test-time optimization, LSTM-Affine performs purely feed-forward adaptation: for each incoming sample, it predicts channel-wise scale and shift and immediately modulates the features—no batch moments, moving averages, or test-time backpropagation are needed. The design is drop-in: in convolutional networks, we place the module after each convolutional block and before activation; in fully connected architectures, we place it before the final classifier.

Our empirical study targets two regimes where non-i.i.d. effects are prominent: (i) FSL on Omniglot, MiniImageNet, and TieredImageNet [2–4], and (ii) SFDA on digits (MNIST, USPS, and SVHN) and Office-31. For SFDA, we adopt a unified SHOT protocol to isolate the contribution of the normalization/affine component: the adaptation pipeline is kept fixed, and we only swap the BN-based affine part for different variants, including our LSTM-Affine. This organization enables like-for-like comparisons while avoiding confounds due to protocol changes. We also relate LSTM-Affine to batch-statistics-free designs that learn affine modulation without explicit whitening [5,7], and to recurrent normalization paradigms that insert temporal modeling into normalization [8]; in contrast, we produce normalization-free, temporally conditioned affine parameters expressly for test-time robustness. In this paper, non-i.i.d. refers to test conditions that deviate from the i.i.d. training assumption, including domain shift (as in SFDA), temporally correlated or streaming inputs, few-shot episodic evaluation with very small or imbalanced batches, and single-instance/small-batch inference where batch statistics are unreliable.

The main contributions of this work are as follows: (1) LSTM-Affine: We introduce a batch-statistics-free, memory-driven affine transformation that replaces BN's affine part and predicts per-instance (γ, β) from a temporal feature context via a lightweight LSTM [6]. (2) Drop-in rule and procedures: We provide a simple integration rule (after each conv block and before activation) and formalize the training and SFDA test-time inference workflows (Algorithms 1 and 2 in Section 3). (3) Unified evaluation and gains: Under a unified SHOT protocol for SFDA—and on standard FSL benchmarks [2–4]—LSTM-Affine consistently improves over BN and adaptive baselines, while requiring no test-time backprop or batch statistics. (4) Analysis of temporal memory: We analyze hidden-state design and reset policies, showing how temporal memory improves stability and robustness under distribution shifts.

2. Related Work

Batch-normalization-based approaches normalize activations using batch moments and then apply a learnable affine transform, yielding faster training and improved generalization [1]. However, their dependency on reliable batch statistics can be problematic under non-i.i.d. or low-batch scenarios, especially at test time. Here, non-i.i.d. refers to test conditions that deviate from the i.i.d. training assumption, including domain shift (e.g., SFDA), temporally correlated streams/online inference, few-shot episodic evaluation with small or imbalanced batches, and single-instance or small-batch inference where

batch statistics are unreliable. Alternatives such as Group Normalization (GN), Instance Normalization (IN), and Layer Normalization (LN) reduce batch dependence but do not explicitly exploit temporal continuity [9–11].

A complementary line removes explicit whitening and instead adapts normalization parameters through conditional modulation, such as FiLM [12] and AdaBN [13]. AdaBN replaces the source-domain batch statistics in BN with those computed from target test data, which still requires batch-level moments [5,7,14] at test time—making it incompatible with instance-wise or online adaptation. Yet these designs often overlook cross-sample temporal structure, limiting adaptation to evolving distributions. Recurrent or memory-based mechanisms introduce temporal modeling into normalization/modulation modules—e.g., Recurrent Batch Normalization—to capture long-term dependencies and maintain adaptation states for continual or online learning [8,15]. Nevertheless, many such methods still rely on explicit statistical normalization, making them sensitive when accurate moment estimates are unavailable.

Meta-learning has also been used to adapt normalization behavior across tasks and domains. MetaBN [2], for instance, meta-parameterizes statistics or affine parameters to improve transfer. Our earlier MetaAFN [16] removes the dependence on batch moments and generates input-adaptive affine parameters from the current instance, but it does not explicitly model temporal memory. Beyond these normalization-centric directions, test-time adaptation (TTA) methods have emerged; a representative BN-based approach is TENT, which minimizes prediction entropy at test time and updates BN-related parameters via backpropagation [17]. In our evaluations, we focus on a unified SHOT protocol for SFDA and isolate the effect of the normalization/affine component by swapping BN for different variants—keeping the rest of the pipeline unchanged—to enable like-for-like comparison.

3. Method: LSTM-Affine

Building on the limitations identified in prior normalization and feature modulation methods, we propose LSTM-Affine, a batch-statistics-free, memory-driven affine transformation mechanism that integrates temporal context into parameter generation.

In this section, we introduce LSTM-Affine, a novel approach designed to replace traditional batch normalization (BN) with a memory-based affine transformation module. The key idea is to eliminate the dependency on batch statistics by learning a dynamic function that generates affine parameters conditioned on the temporal context of input features. This is particularly useful in scenarios such as test-time adaptation, online inference, or few-shot learning, where reliable batch-level normalization is either infeasible or suboptimal.

LSTM-Affine leverages the sequential modeling capability of Long Short-Term Memory (LSTM) [6] networks to maintain a hidden memory state that accumulates information from previously seen features. Instead of normalizing the input using batch-level mean and variance, we directly apply an LSTM-predicted affine transformation to the incoming feature map, thereby achieving the same representational modulation effect while remaining independent of batch composition.

We will now describe the architecture and mechanisms of LSTM-Affine in detail, starting with a conceptual overview, followed by the design of the LSTM-based affine generator, training objectives, and comparisons with traditional BN-based approaches.

3.1. LSTM-Affine: Overview and Architecture

We propose LSTM-Affine, a batch-statistics-free, memory-driven affine transformation that replaces the fixed affine part of batch normalization (BN) with parameters generated from temporal context. Unlike BN—which relies on mini-batch moments followed by a learnable affine transform—LSTM-Affine directly predicts channel-wise scale and shift

for each incoming sample via a lightweight LSTM, enabling per-instance, feed-forward adaptation without batch statistics, moving averages, or test-time backpropagation. In convolutional networks, the module is inserted after each convolutional block and before the activation; in fully connected architectures, it is placed before the final classifier layer. This placement keeps downstream nonlinearities unchanged while allowing the affine modulation to reshape intermediate features analogously to BN's affine step but without requiring batch moments. Each target layer is paired with its own LSTM-Affine submodule and maintains independent recurrent states so that temporal information can be captured locally per depth. Unless otherwise specified, we set the hidden size to $d = 128$ across experiments, which balances adaptation capacity and efficiency.

Let $x_t \in R^{C\times H\times W}$ denote the channel-wise feature map at time t. The module predicts channel-wise scale and shift parameters $\gamma_t, \beta_t \in R^C$ and applies a feature-wise affine multiplication, as shown in (1), where $\odot$ denotes channel-wise multiplication.

For comparison, standard batch normalization (BN), defined in (2), can be reformulated into an equivalent affine form as in (3). In this formulation, μ and σ^2 denote the batch mean and variance, ε is a small constant for numerical stability, and γ, β are learnable affine parameters. The resulting γ' and β' represent the induced affine parameters. This reformulation highlights that BN essentially modulates input features through affine scaling and shifting, with parameters derived from batch statistics—making it a form of statistically driven affine transformation.

In contrast, our LSTM-Affine module retains the affine modulation structure in (1), but eliminates the reliance on batch moments. Instead, it generates the affine parameters (γ_t, β_t) dynamically from the temporal context, enabling more adaptive and sequence-aware modulation.

$$\hat{x}_t = \gamma_t \odot x_t + \beta_t \tag{1}$$

$$BN(x) = \frac{x}{\sqrt{\sigma^2+\varepsilon}} * \gamma + \left(-\frac{\mu}{\sqrt{\sigma^2+\varepsilon}} * \gamma + \beta\right) \tag{2}$$

$$BN(x) = \gamma' * x + \beta', \text{where } \gamma' = \frac{\gamma}{\sqrt{\sigma^2+\varepsilon}},\ \ \beta' = -\frac{\mu}{\sqrt{\sigma^2+\varepsilon}} * \gamma + \beta \tag{3}$$

To enable context-aware affine modulation, we generate the scale and shift parameters using a lightweight LSTM. Given the input feature map x_t, we first apply global average pooling (GAP) to obtain a compact channel descriptor $\overline{x}_t$, as shown in (4). This descriptor $\overline{x}_t$ is then fed into an LSTM that maintains temporal states (h_{t-1}, c_{t-1}) and produces updated hidden and cell states (h_t, c_t) , as shown in (5).

$$\overline{x}_t = \frac{1}{H\cdot W}\sum_{i=1}^{H}\sum_{j=1}^{W} x_t[:, i, j] \tag{4}$$

$$h_t, c_t = LSTM(\overline{x}_t, h_{t-1}, c_{t-1}; \Theta) \tag{5}$$

We then use a linear projection with parameters $W_p \in R^{2C\times d}$ and $b_p \in R^{2C}$ to map the hidden state to the affine parameters, as shown in (6). The resulting modulation is applied channel-wise to every spatial location, as shown in (7).

$$[\gamma_t, \beta_t] = W_p h_t + b_p \tag{6}$$

$$\hat{x}_t[:, i, j] = \gamma_t \odot x_t[:, i, j] + \beta_t \tag{7}$$

Training proceeds end-to-end together with the backbone using the task loss, and no special regularizers beyond standard weight decay are required. At inference, test samples (or mini-batches) are processed sequentially; recurrent states are carried over across samples—or reset according to a specified policy (episode-wise or batch-wise)—to

encode temporal context and track gradual distribution shifts, thereby enabling purely feed-forward test-time adaptation without computing or updating batch statistics and without any optimization at test time. Conceptually, LSTM-Affine can be seen as retaining BN's beneficial affine modulation while replacing batch-moment estimation with a temporally conditioned parameter generator; empirically, the temporal memory stabilizes predictions under continuous distribution shifts and alleviates the fragility of batch statistics in non-i.i.d. or low-batch regimes (e.g., FSL, SFDA, and streaming/online settings).

3.2. Network Design

Following the formulation and core ideas introduced in Section 3.1, this section focuses on the practical architecture and implementation details of the proposed LSTM-Affine module, particularly how it integrates into deep neural networks for adaptive feature modulation.

Each LSTM-Affine unit is inserted into the network as a standalone module assigned to a specific layer—typically after a convolutional block and before the activation function. Rather than applying batch-based normalization followed by a fixed affine transformation, the LSTM-Affine module dynamically generates the affine parameters based on the temporal context encoded in an LSTM. This design allows the model to adapt feature distributions across time, especially in settings where input data are non-i.i.d., such as streaming or episodic test-time scenarios.

To process the input, the feature map is first compressed into a compact representation through global average pooling, producing a channel-wise descriptor that serves as the input to the LSTM. The LSTM maintains its own hidden and cell states across time, capturing contextual patterns and enabling sequential awareness. The output hidden state is then projected through a fully connected layer to produce the channel-wise scale and shift parameters. These parameters are used to perform affine modulation directly on the original feature map, effectively reshaping it in a context-aware manner.

The entire operation is visualized in Figures 1–3. Figure 1 illustrates the internal structure of the LSTM cell, which governs how the current input descriptor interacts with the memory from previous inputs. Figure 2 shows the overall architecture of the affine parameter generator, including how the input descriptor is processed and projected. Figure 3 provides an end-to-end view of the complete data flow: the feature map is pooled, passed through the LSTM-Affine module, and modulated with generated parameters—all achieved without computing any normalization statistics.

By assigning a separate LSTM-Affine generator to each layer, the model can maintain independent temporal memory across network depth. This prevents interference between semantically different layers and improves adaptation fidelity. During training, the modules are updated jointly with the backbone using the same task objective, requiring no additional supervision. At inference, test samples are processed sequentially, and the LSTM states are propagated across inputs—enabling efficient, purely feed-forward adaptation without test-time optimization or backpropagation. This makes the method well-suited for real-time or resource-constrained scenarios where batch normalization and online learning are impractical.

The precise training and test-time procedures are detailed in Algorithm 1 and Algorithm 2, respectively.

The complete operation is summarized in Figure 3: the feature map is pooled into $\overline{x}_t$, processed by the LSTM-APG using both the current descriptor and its memory states, projected into γ_t and β_t, and applied to the feature map. Each LSTM-APG is assigned to a specific layer, maintaining its own memory to avoid interference between semantically distinct features. Training is performed end-to-end with the same task loss as the baseline. During inference, test samples are processed sequentially, carrying forward LSTM states to

adapt to evolving input distributions without relying on batch statistics, moving averages, or backpropagation. For clarity, the pooling step is omitted in Figures 2 and 3, but is applied in all experiments.

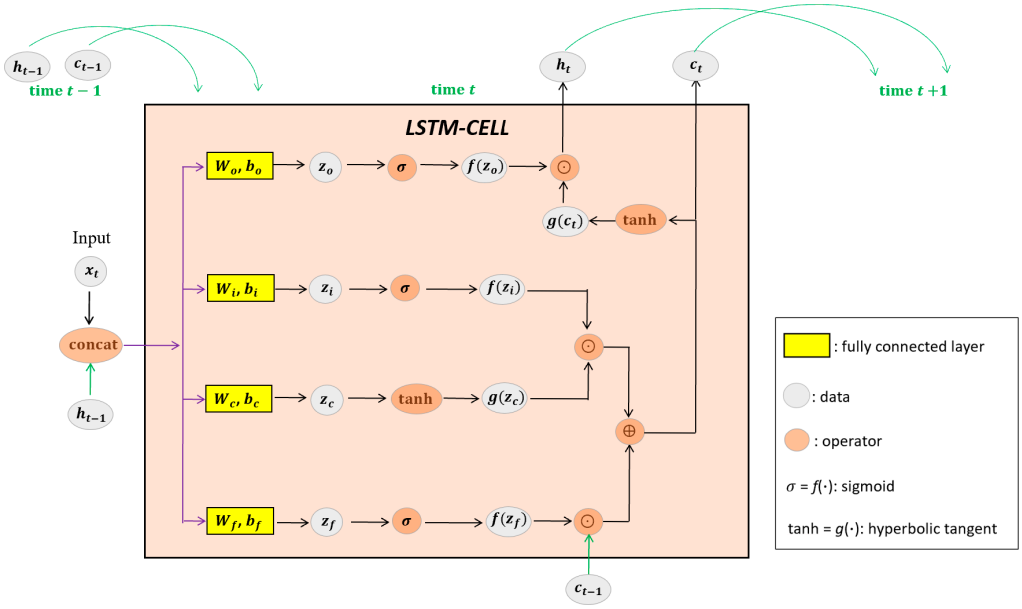


Figure 1. Internal structure of a single LSTM cell used in the LSTM-APG. It receives the pooled feature descriptor, maintains temporal memory via cell and hidden states, and outputs context-aware representations used to generate affine parameters.

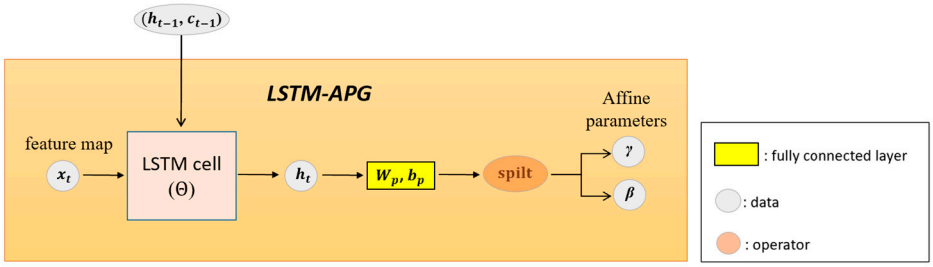


Figure 2. Architecture of the proposed LSTM-APG. The pooled feature descriptor is processed by the LSTM, projected into scale γ_t and shift β_t, and applied to modulate the original feature map.

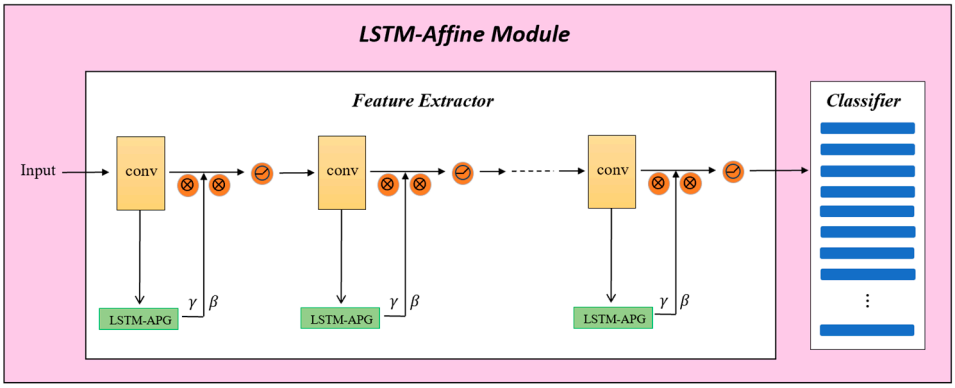


Figure 3. Overall operation of the LSTM-Affine module. The input feature map is globally pooled, processed by the LSTM-APG with temporal context from previous samples, and modulated by the generated affine parameters without explicit normalization.

Implementation details and the step-by-step procedures are provided in Algorithm 1 (training) and Algorithm 2 (SFDA inference).

Algorithm 1: LSTM-Affine—Training/Forward Integration (for FSL and supervised training)

Input: Training set D_{tr}, validation set D_{val}; feature extractor θ_F with L target layers; LSTM-Affine generators $\{\theta_{\text{LSTM},i}\}_{i=1}^{L}$ (output γ, β); classifier θ_C; epochs T; mini-batch size m; learning rate η; hidden size d; state reset policy (when to reset $\{h_i\}$).

Output: Trained parameters θ_F^*, $\left\{\theta_{LSTM,i}^*\right\}_{i=1}^{L}$, θ_C^*.

for epoch $t \leftarrow 1$ to T **do**
  Sample a mini-batch $B_m = \left\{\left(x_j, y_j\right)\right\}_{j=1}^{m} \subset D_{tr}$;
  Apply the state reset policy to initialize/keep hidden states $\{h_i\}_{i=1}^{L}$;
  $Z \leftarrow B_m$; # current tensor traveling through the network
  for layer $i \leftarrow 1$ to L **do**
    $U_i \leftarrow \theta_{F,i}(Z)$; # backbone convolution/FC block output
    $(\gamma_i, \beta_i, h_i) \leftarrow \theta_{\text{LSTM},i}(GAP(U_i), h_i)$; # temporal parameter generation
    $U_i \leftarrow \gamma_i \odot U_i + \beta_i$; # channel-wise affine modulation (no batch statistics)
    $Z \leftarrow Activation(U_i)$ # e.g., ReLU (follow the backbone design)
  end for
  $\hat{y} \leftarrow \theta_C(Z)$;
  $L \leftarrow CE(\hat{y}, y)$; # compute loss
  Update $\theta_F, \{\theta_{\text{LSTM},i}\}_{i=1}^{L}, \theta_C$ by backprop with learning rate η;
  Validate on D_{val} and keep the best checkpoint
end for
return θ_F^*, $\left\{\theta_{LSTM,i}^*\right\}_{i=1}^{L}$, θ_C^*

Algorithm 2: LSTM-Affine—SFDA/SHOT-Style Test-Time Inference (No Backprop)

Input: Pretrained feature extractor θ_F with L target layers; classifier θ_C; LSTM-Affine generators $\{\theta_{\text{LSTM},i}\}_{i=1}^{L}$; unlabeled target-domain stream X_t (single instance or mini-batches); hidden size; state reset policy (episode-/batch-wise).

Output: Target predictions $\{\hat{y}_t\}$ (and carried hidden states).

Initialize/keep hidden states $\{h_i\}_{i=1}^{L}$ according to the reset policy;
for each incoming target sample or mini-batch $x_t \in X_t$ **do**
  $Z \leftarrow x_t$;
  for layer $i \leftarrow 1$ to L **do**
    $U_i \leftarrow \theta_{F,i}(Z)$; # backbone block output
    $(\gamma_i, \beta_i, h_i) \leftarrow \theta_{\text{LSTM},i}(GAP(U_i), h_i)$; # temporal parameter generation
    $U_i \leftarrow \gamma_i \odot U_i + \beta_i$; # channel-wise affine modulation (no batch stats)
    $Z \leftarrow Activation(U_i)$ # e.g., ReLU (follow backbone)
  end for
  $\hat{y}_t \leftarrow arg\ max_k \theta_C(Z)[k]$ # feed-forward prediction (no test-time backprop)
end for

4. Experimental Results

We evaluate the proposed LSTM-Affine on few-shot learning (FSL) and source-free domain adaptation (SFDA) benchmarks, including digit recognition and Office-31 datasets. Unless otherwise noted, results are averaged over five independent runs. Ablation studies

are additionally conducted to analyze the contributions of temporal memory and hidden states.

4.1. Datasets and Experimental Settings

For the FSL experiments, we use three standard benchmarks: Omniglot [18], MiniImageNet [19], and TieredImageNet [20], following the 5-way 5-shot classification protocol [3]. Each model is trained on the respective training split and evaluated on unseen classes from the test split.

For SFDA, we adopt two settings: (1). Digits SFDA: MNIST [21], USPS [22], and SVHN [23], with domain adaptation across U $\rightarrow$ M, S $\rightarrow$ M, and M $\rightarrow$ U tasks. (2). Office-31 SFDA: Amazon (A), Webcam (W), and DSLR (D) domains [24], with adaptation across A $\rightarrow$ D, A $\rightarrow$ W, W $\rightarrow$ D, W $\rightarrow$ A, D $\rightarrow$ A, and D $\rightarrow$ W tasks.

For LSTM-Affine, the LSTM hidden size d is set to 128, determined empirically to balance adaptation capacity and efficiency. A sensitivity study for $d \in \{64, 128, 256\}$ shows minor variation in performance (see also [2,16]). Each LSTM-Affine module is inserted after each convolutional block and before the activation (or before the final classifier in fully connected networks), and maintains its own hidden and cell states (h_i, c_i) per layer. These states are carried across samples to support temporal adaptation during test time, unless explicitly reset in ablation experiments.

4.2. Few-Shot Learning

We first evaluate LSTM-Affine on the 5-way 5-shot classification task. Table 1 compares our method against BN, MetaBN [2], and MetaAFN [16]. The results show that LSTM-Affine, being batch-statistics-free and memory-driven, achieves the best accuracy, surpassing MetaAFN by 2–3%. It is also worth noting that the reported accuracy on Omniglot is much higher than on MiniImageNet and TieredImageNet. This is expected and consistent with prior studies, since Omniglot contains simple handwritten characters with clear structural patterns, making it easier for models to generalize in few-shot settings. In contrast, MiniImageNet and TieredImageNet consist of natural images with high intra-class variability and fine-grained distinctions, which present greater challenges. The performance gap therefore reflects the intrinsic differences in dataset difficulty rather than limitations of the proposed method.

Table 1. Experimental results (%) on the 5-way 5-shot few-shot learning (FSL) tasks. Each value is the average accuracy over five runs.

Method	Omniglot	MiniImageNet	TieredImageNet	Avg.
BN	94.8	67.5	72.4	79.4
MetaBN	95.2	70.2	75.5	80.3
MetaAFN	96.7	72.3	77.0	82.0
LSTM-Affine (ours)	99.0	74.5	79.4	84.3

4.3. Source-Free Domain Adaptation (Digits)

We further evaluate our method on digit classification with domain shifts (MNIST, USPS, and SVHN). Following the clarification in Section 4.1, all SFDA baselines in Tables 2 and 3 are implemented within the same SHOT protocol, differing only in the normalization/affine component. Table 2 reports average accuracy across all domain pairs. LSTM-Affine consistently outperforms all baselines. Unlike optimization-based test-time adaptation methods such as SHOT, which require backpropagation during inference and incur additional computational and memory overhead, LSTM-Affine performs purely feed-

forward adaptation through its recurrent generator, maintaining efficiency comparable to standard BN inference.

Table 2. Comparison of source-free domain adaptation (SFDA) results (%) on digit classification tasks with domain shifts among USPS, MNIST, and SVHN. Each method is evaluated using the SHOT protocol, and results are averaged over all domain pairs.

Method	U → M	S → M	M → U	Avg.
SHOT (BN)	97.71	99.05	98.17	98.31
SHOT (MetaBN)	97.66	98.82	98.17	98.22
SHOT (MetaAFN)	97.61	99.09	98.17	98.29
SHOT (LSTM-Affine)	98.04	99.10	98.66	98.60

Table 3. Comparison of source-free domain adaptation (SFDA) results (%) on the Office-31 dataset across three transfer tasks under the SHOT protocol. Reported values are average accuracies over five runs.

Method	A → D	A → W	W → D	W → A	D → A	D → W	Average
SHOT (BN)	90.29	92.24	99.90	72.82	73.39	97.78	87.74
SHOT (MetaBN)	92.21	92.20	99.80	71.14	73.77	98.03	87.86
SHOT (MetaAFN)	93.52	90.25	99.60	76.75	75.21	98.10	88.91
SHOT (LSTM-Affine)	94.05	92.26	99.90	77.94	76.14	98.37	89.78

4.4. Source-Free Domain Adaptation (Office-31)

We also evaluate our method on the Office-31 dataset (A → D, A → W, W → D, W → A, D → A, D → W). Table 3 shows that LSTM-Affine achieves the best average accuracy among all compared methods.

4.5. Ablation Study

To better understand the roles of temporal memory and hidden states in LSTM-Affine, we perform ablation experiments by selectively removing these components (Table 4). Resetting the LSTM for each sample ("No memory") prevents the model from accumulating contextual information, which slows adaptation to gradual distribution shifts and increases sensitivity to abrupt changes. Eliminating the hidden state ("No hidden state") removes short-term dependency, resulting in noisier affine parameters and less stable predictions. Both ablations lead to performance degradation relative to the full model, yet still outperform MetaAFN. This observation highlights that recurrent modeling—even when partially disabled—provides more effective adaptation than relying solely on meta-learning, while maintaining the batch-statistics-free nature of the framework.

Table 4. Ablation study of LSTM-Affine. Average accuracy (%) across all benchmarks when removing temporal memory ("No memory") or hidden state ("No hidden state"). Both ablations reduce performance compared to the full model, yet still outperform MetaAFN, indicating that recurrent modeling—even when partially disabled—offers stronger adaptation than meta-learning alone.

Method	Few-Shot	Digits SFDA	Office-31 SFDA	Average	Gap to Full
Full	84.3	98.6	89.8	90.9	—
No memory	83.1	98.2	89.3	90.2	−0.7

Table 4. *Cont.*

Method	Few-Shot	Digits SFDA	Office-31 SFDA	Average	Gap to Full
No hidden state	83.5	98.3	89.4	90.4	−0.5
MetaAFN	82.0	98.3	88.9	89.7	−1.2

5. Conclusions and Future Work

This paper introduced LSTM-Affine, a memory-based affine transformation module that serves as a drop-in replacement for batch normalization (BN) in deep neural networks. By leveraging an LSTM conditioned on historical input features, the module dynamically generates scale and shift parameters without relying on batch-level statistics or moving averages. This batch-statistics-free design enables robust adaptation to distributional shifts in settings where conventional normalization is unreliable, such as single-instance, streaming, or test-time inference.

Extensive experiments on few-shot learning and source-free domain adaptation benchmarks—including Omniglot, MiniImageNet, TieredImageNet, digit datasets, and Office-31—demonstrated that LSTM-Affine consistently outperforms or matches strong baselines such as BN and MetaBN. The method achieves competitive accuracy even under severe domain shifts, while maintaining efficiency by avoiding test-time backpropagation.

Beyond accuracy, LSTM-Affine offers architectural simplicity, full differentiability, and temporal awareness through its built-in memory mechanism, making it a compelling alternative to traditional normalization layers. Future work will investigate meta-learning-based training strategies, such as episodic optimization or gradient-based meta-updates, to further improve adaptability to unseen domains and extend applicability to continual learning scenarios.

Author Contributions: Conceptualization, J.P.Y., J.-M.F. and H.J.L.; Methodology, J.-M.F.; Investigation, J.P.Y., H.J.L. and Y.T.; Resources, Y.T.; Writing—Original Draft, J.P.Y. and H.J.L.; Writing—Review and Editing, J.-M.F. and Y.T.; Visualization, Y.T.; Supervision, J.P.Y., J.-M.F. and H.J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available in public domain repositories. These datasets were obtained from the following sources: Omniglot (https://github.com/brendenlake/omniglot, accessed on 10 October 2025), TieredImageNet (https://github.com/renmengye/few-shot-ssl-public, accessed on 10 October 2025), USPS (https://www.kaggle.com/datasets/bistaumanga/usps-dataset, accessed on 10 October 2025), SVHN (http://ufldl.stanford.edu/housenumbers/, accessed on 10 October 2025), MiniImageNet (https://www.kaggle.com/datasets/arjunashok33/miniimagenet, accessed on 10 October 2025), MNIST (https://www.kaggle.com/datasets/hojjatk/mnist-dataset, accessed on 10 October 2025), Office-31 (https://www.kaggle.com/datasets/xixuhu/office31, accessed on 10 October 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning (ICML), Lille, France, 6–11 July 2015; pp. 448–456.
2. Gao, W.; Zhou, J.; Metaxas, D.; Wang, J. Meta-Batch Normalization for Few-Shot Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 10195–10204.
3. Snell, J.; Swersky, K.; Zemel, R. Prototypical networks for few-shot learning. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17), Long Beach, CA, USA, 4–9 December 2017; pp. 4080–4090.

4. Khodadadeh, S.; Boloni, L.; Shah, M. Unsupervised meta-learning for few-shot image classification. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 8 December 2019; pp. 10132–10142.
5. Santurkar, S.; Tsipras, D.; Ilyas, A.; Madry, A. How Does Batch Normalization Help Optimization? *arXiv* **2018**, arXiv:1805.11604v5.
6. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
7. Brock, A.; De, S.; Smith, S.L.; Simonyan, K. High-performance large-scale image recognition without normalization. In Proceedings of the International Conference on Machine Learning (ICML), Virtual, 18–24 July 2021; pp. 1059–1071.
8. Cooijmans, T.; Ballas, N.; Laurent, C.; Gülçehre, C.; Courville, A. Recurrent Batch Normalization. *arXiv* **2016**, arXiv:1603.09025v5.
9. Wu, Y.; He, K. Group normalization. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
10. Ulyanov, D.; Vedaldi, A.; Lempitsky, V. Instance normalization: The missing ingredient for fast stylization. *arXiv* **2016**, arXiv:1607.08022v3.
11. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer normalization. *arXiv* **2016**, arXiv:1607.06450v1. [CrossRef]
12. Perez, E.; Strub, F.; de Vries, H.; Dumoulin, V.; Courville, A. FiLM: Visual reasoning with a general conditioning layer. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
13. Li, Y.; Wang, N.; Shi, J.; Liu, J.; Hou, X. Revisiting Batch Normalization for Practical Domain Adaptation. *arXiv* **2016**, arXiv:1603.04779v4. [CrossRef]
14. Dumoulin, V.; Perez, E.; Schucher, N.; Strub, F.; Vries, H.D.; Courville, A. Feature-Wise Transformations. Distill. 2018. Available online: https://distill.pub/2018/feature-wise-transformations/ (accessed on 21 October 2025).
15. Yoon, J.; Kim, T.; Kim, D.; Kim, S.; Bengio, Y. Learning to modulate for memory-augmented few-shot learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 13684–13693.
16. Yeh, J.P.; Lin, H.J.; Tsai, Y.; Tokuyama, Y.; Hsu, W.-L. Meta Affine Transformation: A Batch-Statistics-Free Adaptive Normalization Method for Robust Few-Shot Learning and Domain Adaptation. *Int. J. Pattern Recognit. Artif. Intell.* 2025, *in press*.
17. Wang, Y.; Wang, Q.; Bai, S.; Yuille, A.L.; Gao, T. Tent: Fully test-time adaptation by entropy minimization. In Proceedings of the International Conference on Learning Representations (ICLR), Vienna, Austria, 4 May 2021.
18. Lake, B.M.; Salakhutdinov, R.; Tenenbaum, J.B. Human-level concept learning through probabilistic program induction. *Science* **2015**, *350*, 1332–1338. [CrossRef] [PubMed]
19. Vinyals, O.; Blundell, C.; Lillicrap, T.; Kavukcuoglu, K.; Wierstra, D. Matching networks for one shot learning. *arXiv* **2016**, arXiv:1606.04080v2. [CrossRef]
20. Ren, M.; Triantafillou, E.; Ravi, S.; Snell, J.; Swersky, K.; Tenenbaum, J.B.; Larochelle, H.; Zemel., R.S. Meta-learning for semi-supervised few-shot classification. In Proceedings of the International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
21. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
22. Hull, J.J. A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.* **1994**, *16*, 550–554. [CrossRef]
23. Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; Ng, A.Y. Reading digits in natural images with unsupervised feature learning. In Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning, Granada, Spain, 12–17 December 2011.
24. Saenko, K.; Kulis, B.; Fritz, M.; Darrell, T. Adapting visual category models to new domains. In Proceedings of the 11th European Conference on Computer Vision (ECCV), Heraklion, Greece, 5–11 September 2010; pp. 213–226.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.