Continual Test-time Adaptation for End-to-end Speech Recognition on Noisy Speech

Guan-Ting Lin⋆, Wei-Ping Huang⋆, Hung-yi Lee
Graduate Institute of Communication Engineering, National Taiwan University
Taiwan
{f10942104, hungyilee}@ntu.edu.tw, thomas1232121@gmail.com

Abstract

Deep Learning-based end-to-end Automatic Speech Recognition (ASR) has made significant strides but still struggles with performance on out-of-domain samples due to domain shifts in real-world scenarios. Test-Time Adaptation (TTA) methods address this issue by adapting models using test samples at inference time. However, current ASR TTA methods have largely focused on non-continual TTA, which limits cross-sample knowledge learning compared to continual TTA. In this work, we first propose a Fast-slow TTA framework for ASR that leverages the advantage of continual and non-continual TTA. Following this framework, we introduce Dynamic SUTA (**DSUTA**), an entropy-minimization-based continual TTA method for ASR. To enhance DSUTA robustness for time-varying multi-domain data, we design a **dynamic reset strategy** to automatically detect domain shifts and reset the model. Our method demonstrates superior performance on various noisy ASR datasets, outperforming both non-continual and continual TTA baselines while maintaining robustness to domain changes without requiring domain boundary information[1].

1 Introduction

Deep learning-based end-to-end Automatic Speech Recognition (ASR) has made remarkable progress in recent years, achieving low recognition error rates for in-domain samples. However, domain shifts frequently occur in real-world scenarios. Although recent large-scale ASR models exhibit some generalization to out-of-domain test samples, their performance on out-of-domain samples still lags behind the in-domain performance.

Test-time adaptation (TTA) is an attractive method to address domain shift issues during inference time. TTA adapts the model using only single or batched test samples without needing the source training data at testing time. Specifically, the source model is adapted via unsupervised objectives like Entropy Minimization (EM) (Wang et al., 2020) or Pseudo-Labeling (PL) (Goyal et al., 2022) in inference time. TTA methods can be characterized into two categories: **1) Non-continual TTA** methods adapt the source model for each test utterance and reset to the original model for subsequent samples (Wang et al., 2020), and **2) Continual TTA** (CTTA) continuously adapts the model for target domains, leveraging knowledge learned across samples to improve performance (Niu et al., 2022a,b; Press et al., 2024).

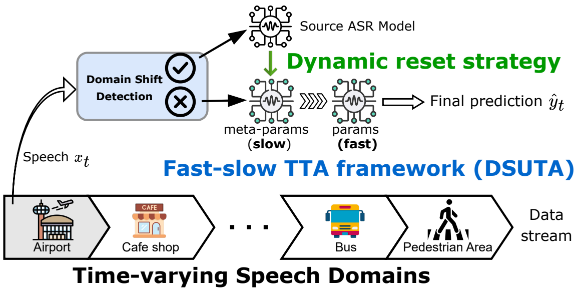

Figure 1: Illustration of the proposed Fast-slow TTA framework and dynamic reset strategy with time-varying speech domains. The Fast-Slow TTA framework includes meta-parameters that update *slowly* to capture cross-domain knowledge, while other parameters update *fast* for the incoming test samples. The **Dynamic reset strategy** automatically detects domain shifts and resets the model to the source model.

TTA methods initially strive in the field of computer vision (Wang et al., 2020; Niu et al., 2022a,b; Press et al., 2024). In speech recognition, recent studies have tailored TTA methods with EM-based optimization (Lin et al., 2022; Kim et al., 2023; Liu et al., 2023), proposing new training objectives and demonstrating effectiveness across datasets. However, existing ASR TTA methods only focus on non-continual TTA, constraining the

[1] The source code is available at https://github.com/hhhaaahhhaa/Dynamic-SUTA

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 20003–20015
November 12-16, 2024 ©2024 Association for Computational Linguistics

model to learn knowledge across samples. There is limited research on CTTA for end-to-end ASR. Recently, AWMC (Lee et al., 2023) proposed a pseudo-labeling CTTA method for ASR on a single test domain. However, as shown in previous work (Lin et al., 2022), pseudo-labeling is not as effective as EM-based methods, and its ability on long multi-domain testing data is unknown.

In this work, we propose a general **Fast-slow TTA** framework that leverages the advantages of both continual and non-continual TTA. Based on this framework, we introduce an EM-based CTTA method named **D**ynamic **SUTA** (**DSUTA**) for ASR. Furthermore, to enhance the robustness of DSUTA on time-varying domain data, we propose a **dynamic reset strategy** to automatically detect domain shifts and determine when to reset the model to the original source model. This strategy improves Fast-slow TTA over long sequences of multi-domain data streams.

We demonstrate the effectiveness of our method on single-domain and multi-domain time-varying ASR benchmarks under different acoustic conditions, simulating real-world changing environments. Our method outperforms the strong single-utterance baseline SUTA (Lin et al., 2022) and the CTTA baseline AWMC (Lee et al., 2023), showing robustness to domain changes even without knowing the domain boundaries.

Our contributions can be summarized as follows:

1. Propose the Fast-slow TTA framework to bridge the gap between continual and non-continual TTA.

2. Introduce a specific version of the Fast-slow TTA method named **DSUTA** with a novel dynamic reset strategy to stabilize CTTA over multi-domain and long test data streams.

3. Demonstrate significant improvement over both non-continual and continual baselines on single-domain and time-varying data.

2 Related Works

2.1 Non-continual TTA for ASR

Non-continual TTA methods adapt the source model for each test utterance and reset to the original model for subsequent samples. SUTA (Lin et al., 2022) introduces the first TTA approach for non-autoregressive ASR, based on entropy minimization and minimum class confusion. SGEM (Kim et al., 2023) extends TTA to autoregressive ASR models by introducing a general form of entropy minimization. Liu et al. (2023) enhances TTA with confidence-enhanced entropy minimization and short-term consistency regularization. However, these non-continual TTA methods view each utterance independently, which only relies on a single utterance and fails to leverage the knowledge across a stream of test samples to improve the adaptation.

2.2 Continual TTA

Unlike non-continual TTA, which resets to the source model for each sample, continual TTA enables the online model to use learned knowledge to handle gradual changes in the target domain. However, it may suffer from model collapse if adaptation is unstable when the data stream is too long. To improve the performance and stability of CTTA, studies in the computer vision field have developed solutions like stochastic model restoring (Wang et al., 2022), sample-efficiency entropy minimization (Niu et al., 2022a), sharpness-aware reliable entropy minimization (Niu et al., 2022b), and fixed frequency model reset (Press et al., 2024).

In the ASR research, *there are limited studies on CTTA ASR*. Recently, AWMC (Lee et al., 2023) attempts continual TTA on ASR using a pseudo-labeling approach with an extra anchor model to prevent model collapse. However, AWMC (Lee et al., 2023) only measures the performance on single-domain data with the pseudo-labeling method. This work focuses on multi-domain time-varying long data streams. We propose a fast-slow TTA framework and dynamic reset strategy based on an entropy minimization-based CTTA method, which achieves better performance and stability.

3 Methodology

Section 3.1 describes the proposed **Fast-slow TTA framework**. Following this framework, Section 3.2 extends SUTA into **Dynamic SUTA**. To handle multi-domain scenarios better, we propose a **dynamic reset strategy** in Section 3.3.

3.1 Fast-slow TTA Framework

Non-continual TTA treats each sample as an independent learning event. The adaptation process can fit the current sample without affecting future samples; however, the learned knowledge cannot be

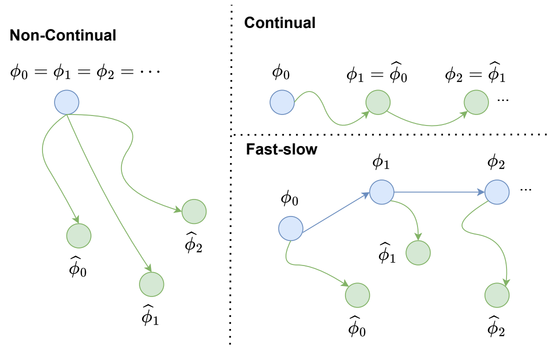

Figure 2: Illustration of the 3 different TTA approaches.

transferred to future samples. In contrast, continual TTA utilizes learned knowledge, but overfitting the current sample can adversarially degrade performance on future samples. For instance, if the model overly fits the current sample and (model collapse), the performance on future samples will significantly degrade with continual TTA, whereas in non-continual TTA, the performance remains unaffected.

We propose **Fast-slow TTA**, a new CTTA framework that leverages learned knowledge while retaining the benefits of non-continual TTA, as shown in Figure 2. Fast-slow TTA aims to learn meta-parameters ϕ_t which evolve slowly over time. Instead of always starting the adaptation process from the pre-trained parameters, as in non-continual TTA, we start from ϕ_t at time step t. Specifically,

$$\begin{aligned} \phi_0 &= \phi_{pre}, \\ \widehat{\phi}_t &= A(\phi_t, x_t), \\ \widehat{y}_t &= \widehat{\phi}_t(x_t), \\ \phi_{t+1} &= U(\phi_t, x_t), \end{aligned}$$

where ϕ_{pre} are the pre-trained parameters, and A and U represent an adaptation algorithm and an update algorithm, respectively. The evaluation is based on the online predictions $\widehat{y}_t$.

The meta-parameters ϕ_t can leverage knowledge across samples. These parameters are slowly updated by U, and the final prediction is made after a fast adaptation A. This allows the parameters to fit the current sample for greater improvement while mitigating the risk of model collapse over time.

Fast-slow TTA generalizes to continual and non-continual TTA. If $U(\phi_t, x_t) = \phi_t$, i.e., ϕ_t remains constant over time, the framework degenerates to non-continual TTA. On the other hand, if $A = U$, i.e. $\phi_{t+1} = \widehat{\phi}_t$, the framework degenerates to continual TTA.

Algorithm 1 Dynamic SUTA

```
Input: Data stream $\{x_t\}_{t=1}^{T}$, buffer $\mathcal{B}$ with size $M$, adaptation step $N$, pre-trained param $\phi_{pre}$
Output: Predictions $\{\widehat{y}_t\}_{t=1}^{T}$
$\mathcal{B}, \phi_1 \leftarrow \{\}, \phi_{pre}$
$Results \leftarrow \{\}$
for $t = 1$ to $T$ do
    $\widehat{\phi}_t \leftarrow \phi_t$                          ▷ Adapt parameters
    for $n = 1$ to $N$ do
        $\mathcal{L} \leftarrow \mathcal{L}_{suta}(\widehat{\phi}_t, x)$
        $\widehat{\phi}_t \leftarrow$ Optimizer($\widehat{\phi}_t, \mathcal{L}$)
    $\widehat{y}_t \leftarrow \widehat{\phi}_t(x_t)$                     ▷ Save prediction
    $Results \leftarrow Results \cup \{\widehat{y}_t\}$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{x_t\}$
    if $t\%M = 0$ then                   ▷ Update meta-param
        $\mathcal{L} \leftarrow \frac{1}{M}\sum_{x \in \mathcal{B}} \mathcal{L}_{suta}(\phi_t, x)$
        $\phi_{t+1} \leftarrow$ Optimizer($\phi_t, \mathcal{L}$)
        $\mathcal{B} \leftarrow \{\}$
    else
        $\phi_{t+1} \leftarrow \phi_t$
return $Results$
```

3.2 Dynamic SUTA

We propose **D**ynamic **SUTA** (DSUTA), a fast-slow TTA method based on SUTA (Lin et al., 2022). Specifically, given pre-trained parameters ϕ_{pre}, for every incoming sample x_t, SUTA adapts ϕ_{pre} for N steps with the objective $\mathcal{L}_{suta}$. $\mathcal{L}_{suta}$ consists of entropy loss and minimum class confusion loss. Entropy minimization aims to sharpen class distribution, and minimum class confusion aims to reduce the correlation between different prediction classes. See Appendix A.4 for the detailed loss function. Model parameters are reset to ϕ_{pre} when the next sample arrives.

For DSUTA, the adaptation algorithm A is set exactly the same as SUTA, which iteratively adapts ϕ_t for N steps with $\mathcal{L}_{suta}$ on x_t. To construct the update algorithm U, we introduce a small buffer $\mathcal{B}$ with size M. For every M step, the buffer is filled and we calculate $\mathcal{L}_{suta}$ from these M samples to update the meta-parameters ϕ_t with gradient descent. The buffer is then cleared. Thus, the meta-parameters ϕ_t gradually evolve by mini-batch gradient descent with batch size M. DSUTA can be viewed as a variant of SUTA, which starts the adaptation from dynamically changing ϕ_t instead of the fixed ϕ_{pre}. Denote $\mathcal{L}_{suta}(\phi, x)$ as the loss of sample x on model ϕ. Algorithm 1 describes the pseudo code of DSUTA.

3.3 DSUTA with Dynamic Reset Strategy

As time progresses and the testing domain changes, multiple domain shifts significantly challenge the robustness of continual TTA methods. Recently, Press et al. (2024) has shown that *model reset* at a fixed frequency, which resets the current parameters to the pre-trained ones at regular intervals, is a simple yet effective strategy. Therefore, we attempt to utilize *model reset* strategy to update the meta-parameters ϕ_t in DSUTA[2]. However, determining the optimal reset frequency in reality is challenging. To automatically determine when to apply model reset to ϕ_t, we propose a **dynamic reset strategy** that actively detects large distribution shifts and dynamically resets $\phi_{t+1} = \phi_{pre}$.

Figure 3 provides an illustration of DSUTA with the dynamic reset strategy. Since distribution shift is a relative concept that is well-defined only after a base domain is constructed, we designed a ***domain construction stage*** and a ***shift detection stage***. Our proposed method alternates between these two stages over time. The domain construction stage first constructs a base domain $\mathcal{D}$ with K samples. No model reset will be applied during this stage. In the subsequent shift detection stage, a detection algorithm checks each incoming sample to determine if there is a significant distribution shift. If a large shift is detected, we apply model reset and switch to a new domain construction stage.

The following subsections describe the strategy in detail. We first introduce the Loss Improvement Index in Section 3.3.1, which measures the extent of the distribution shift. Then we define the domain construction stage and the shift detection stage in Section 3.3.2.

3.3.1 Loss Improvement Index (LII)

We aim to find an indicator that measures the extent of the distribution shift from the base domain $\mathcal{D}$. To identify an appropriate indicator, we observed that given a model $\phi_{\mathcal{D}}$ trained on domain $\mathcal{D}$, $\mathcal{L}_{suta}$ for in-domain samples is empirically lower than that for out-of-domain samples. This suggests that $\mathcal{L}_{suta}(\phi_{\mathcal{D}}, x_t)$ might be a good indicator. Additionally, we found that subtracting the loss from the pre-trained model, $\mathcal{L}_{suta}(\phi_{pre}, x_t)$, is beneficial to normalize the inherent difficulty introduced by the data sample itself[3]. Overall, we define **L**oss **I**mprovement **I**ndex (LII) as our indicator:

$$LII_t = \mathcal{L}(\phi_{\mathcal{D}}, x_t) - \mathcal{L}(\phi_{pre}, x_t),$$

where $\mathcal{L} = \mathcal{L}_{suta}$. The construction of $\phi_{\mathcal{D}}$ will be described in the next section.

3.3.2 Domain Construction Stage and Shift Detection Stage

We integrate DSUTA with the dynamic reset strategy as follows. Assume the model has been reset at time step r.

(1) Domain Construction Stage:

1. Let $k = \lfloor \frac{K}{2} \rfloor$, construct $\phi_{\mathcal{D}} = \phi_{r+k}$.
2. Collect LII_t for $t \in [r+k+1, r+K]$.
3. At the end of the stage (i.e., $t = r + K$), compute $\mathcal{G}_{\mathcal{D}} = \mathcal{N}(\mu, \sigma^2)$ from the collected LIIs.

The goal is to estimate the distribution of LII. We construct $\phi_{\mathcal{D}} = \phi_{r+k}$ as the meta-parameters after observing k samples since the last reset. Calculating the LII requires $\phi_{\mathcal{D}}$, and since TTA is an online process, $K - k$ is the number of LIIs we can collect for statistical estimation. A smaller k might not suffice for $\phi_{\mathcal{D}}$ to adequately represent the domain, while a larger k reduces the number of data points we can gather for estimation. Therefore, we empirically set $k = \lfloor \frac{K}{2} \rfloor$.

(2) Shift Detection Stage:

$$\phi_{t+1} = \begin{cases} \phi_{pre}, & \text{if } \frac{LII_t - \mu}{\sigma} > 2, \\ U_{DSUTA}(\phi_t, x_t), & \text{otherwise,} \end{cases}$$

where U_{DSUTA} is the update algorithm of DSUTA.

During the domain construction stage, we develop a statistical model $\mathcal{G}_{\mathcal{D}}$ using $K - k$ samples to estimate the distribution of LII. In the shift detection stage, we trigger a reset operation if the LII exceeds a certain threshold, indicating an abnormally large shift. To determine whether the LII indicates such a shift, we conduct a right-tailed hypothesis test.

For the right-tailed hypothesis test, the common practice with a significance level of 0.05 corresponds to a Z-score of 1.64. Here, we use a Z-score of 2 for simplicity, which makes the condition for resetting slightly stricter.

Additionally, using the LII of a single sample for the hypothesis test is too sensitive. The **averaged LII** from multiple samples reduces variance and

[2] Non-continual TTA can be viewed as the case where we apply model reset at every time step.

[3] See Section 5.1 for more discussion on indicator choice.

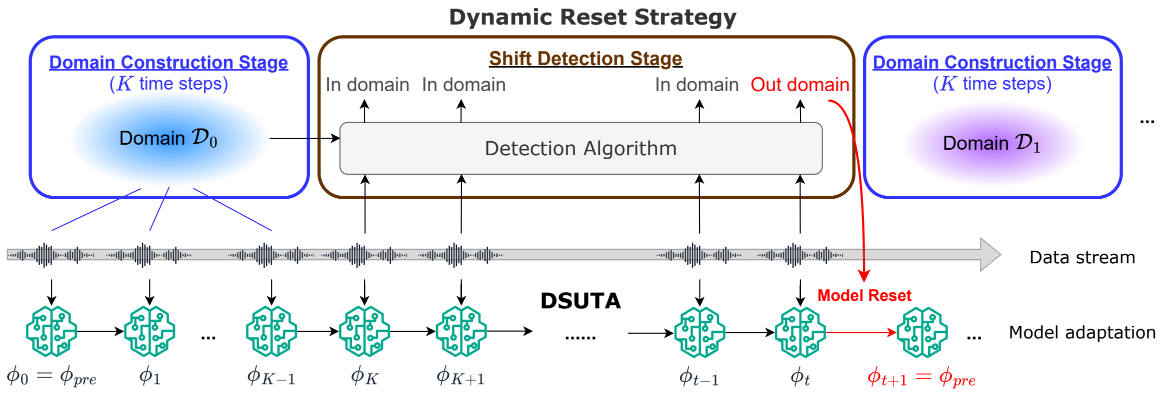

Figure 3: Sketch of DSUTA with the dynamic reset strategy. The domain construction stage and the shift detection stage alternate over time. When a large shift is detected, apply model reset to DSUTA, i.e., update $\phi_{t+1} = \phi_{pre}$.

yields more reliable results. With DSUTA, we perform the hypothesis test every M step, using the M samples in DSUTA's buffer to calculate the averaged LII. The final shift detection stage is defined as follows:

$$\phi_{t+1} = \begin{cases} \phi_{pre}, & \text{if } \frac{1}{M}\sum_{i|x_i\in\mathcal{B}} \frac{LII_i-\mu}{\sigma/\sqrt{M}} > 2, M|t, \\ U_{DSUTA}(\phi_t, x_t), & \text{otherwise.} \end{cases}$$

Here, $\mathcal{B}$ represents the buffer containing the most recent M samples. In our implementation, we further introduce a patience parameter P to enhance the stability. Please refer to the Appendix Algorithm 2 for details.

4 Experiments

4.1 Dataset

4.1.1 Single-domain Simulated Noisy Data

Corrupted Librispeech (LS-C): we follow previous works (Kim et al., 2023) by adding background noises from MS-SNSD (Reddy et al., 2019) into Librispeech test set (Panayotov et al., 2015). The noises include air conditioner (AC), airport announcement (AA), babble (BA), copy machine (CM), munching (MU), neighbors (NB), shutting door (SD), typing (TP), and vacuum cleaner (VC). We also apply Gaussian noise (GS) as in (Lin et al., 2022), resulting in 10 different noises in total. The Signal-to-Noise Ratio (SNR) is set to 5 dB.[4]

4.1.2 Multi-domain Time-varying Data

We create three time-changing multi-domain test data streams by concatenating different corruptions from LS-C.

(a) MD-Easy: Noises in MD-Easy are determined by the relatively *well-performed* noises of the pretrained model (See Table 1). Five background noises, in the order AC→CM→TP→AA→SD, were used, with 500 samples for each noise, making a total of 2500 samples.
(b) MD-Hard: Noises in MD-Hard are determined by the relatively *poor-performed* noises of the pretrained model (See Table 1). Five background noises, in the order GS→MU→VC→BA→NB, were used, with 500 samples for each noise, making a total of 2500 samples.
(c) MD-Long: We first sample a background noise from the 10 available background noises, then sample a data sequence with this noise, with a *random length ranging from 20 to 500*. We repeat this process until the total length reaches 10,000.

4.1.3 Multi-domain Real Noisy Data

CHiME-3 (Barker et al., 2017): a noisy version of WSJ corpus mixed with real speech recorded in four noisy environments (Cafe, Bus, Street, Pedestrian Area). In this work, different types of noisy speech are randomly distributed in a sequence across time.

4.2 Baselines

4.2.1 Non-continual TTA Baselines

1) SUTA (Lin et al., 2022) leverages unsupervised objectives (entropy minimization and minimum class confusion) to reduce uncertainty and minimize class correlations. Temperature smoothing is applied to flatten the output probability distributions, addressing issues with over-confident predictions. The adaptation process involves iteratively optimizing the objective of entropy minimization and minimal class correlation. **2) SGEM (Kim**

[4] (Kim et al., 2023) reported using 10dB noise but their source code and results show that they use 5 dB.

et al., 2023) propose a general form of entropy minimization with negative sampling.

4.2.2 Continual TTA Baselines

3) CSUTA is a straightforward continual version of SUTA without resetting parameters. **4) AWMC** (Lee et al., 2023) utilizes the anchor model to generate initial pseudo labels, the chaser model updates itself using these pseudo labels for self-training, and the leader model refines predictions through an exponential moving average.

4.3 Implementation Details

We use the wav2vec 2.0-base model fine-tuned on Librispeech 960 hours[5] as the source ASR model. For SUTA, we follow the official implementation[6], where an additional reweighting trick is applied on the minimum class confusion loss. The default adaptation step of SUTA is $N = 10$, as specified in the original paper. For SGEM, we follow the official implementation[7]. For CSUTA, we set the adaptation step to $N = 1$ since we found that any higher value would cause severe model collapse. We reimplemented AWMC with wav2vec 2.0, as there is no official code, and all hyperparameters follow the original paper. For the proposed DSUTA, the default buffer size is $M = 5$, and the adaptation step is $N = 10$. To reduce GPU memory usage, we exclude samples with raw lengths longer than 20 seconds in all experiments. This removes about 1% of the data.

For hyperparameter search, we investigate batch sizes (M=3, 5, 10) and domain construction steps (K=50, 100, 200), and find out that our method is robust across different setups. For more details, please see the Appendix A.2 section.

4.4 Results

4.4.1 Single Domain

We compare TTA performance on LS-C by Word Error Rate (WER) in Table 1. DSUTA shows significant improvement compared to the baseline methods. It outperforms both non-continual and continual baseline methods by a large margin, except for the SD domain, where it still achieves a 15.5% WER, close to SGEM's performance (14.9%). Notably, on the NB domain, DSUTA achieves a 36.3% WER compared to SUTA, which has a WER greater than 100%, demonstrating the effectiveness of our method.

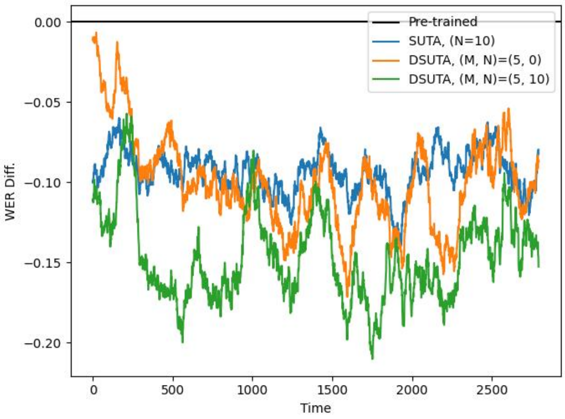

Figure 4: WER difference compared to the pre-trained model on CM domain over time. Data is smoothed by a window with a size of 100.

The key success factor of DSUTA is its ability to leverage learned knowledge from past samples. Figure 4 plots the WER difference compared to the pre-trained model on the CM domain over time. We compare three methods: SUTA with $N = 10$, DSUTA with $(M, N) = (5, 10)$, and DSUTA with $(M, N) = (5, 0)$, i.e., the learned ϕ_t itself. The WER of ϕ_t is lower than that of the pre-trained model, and DSUTA with 10-step adaptation outperforms SUTA with 10-step adaptation. In other words, DSUTA adaptation has a “better start" compared to non-continual TTA methods due to the learned knowledge, resulting in superior performance.

Table 1 also compares other continual TTA methods. Naive continual training, such as CSUTA, results in unsatisfactory performance and is sometimes even worse than the original pre-trained model due to its instability. Although AWMC is designed to increase stability, its performance sometimes lags behind SUTA, particularly in cases where the original pre-trained model has an extremely high error rate (BA and NB). This is not surprising since AWMC relies on a pseudo-label approach. In contrast, DSUTA uses mini-batch gradient descent to enhance stability without the use of pseudo labels. Furthermore, the fast-slow approach allows DSUTA to inherit SUTA's ability to better fit a single utterance, improving overall performance while avoiding the meta-parameters

[5] https://huggingface.co/facebook/wav2vec2-base-960h

[6] https://github.com/DanielLin94144/Test-time-adaptation-ASR-SUTA

[7] https://github.com/drumpt/SGEM

Method	AA	AC	BA	CM	GS	MU	NB	SD	TP	VC	CHiME-3
Source model	40.6	27.7	66.9	49.7	75.6	51.4	120.1	19.4	25.8	49.7	30.0
Non-continual											
SUTA	30.6	17.4	53.7	38.7	54.5	39.0	112.3	15.0	17.4	39.3	23.3
SGEM	30.9	17.8	54.5	39.2	56.3	39.2	113.0	**14.9**	17.5	40.3	23.5
Continual											
CSUTA	39.8	22.6	63.4	53.4	58.4	54.7	68.1	23.2	23.0	50.9	27.6
AWMC	31.6	18.0	61.6	37.7	48.5	36.2	131.9	17.0	18.0	36.1	22.4
Fast-slow											
DSUTA	**25.9**	**15.4**	**33.2**	**33.5**	**37.0**	**28.4**	**36.3**	15.5	**15.6**	**29.9**	**21.7**

Table 1: WER (%) of different TTA methods on LS-C with 10 types of noises and CHiME-3. Reported WER is averaged over 3 runs.

Method	MD-Easy	MD-Hard	MD-Long
Source model	32.7	74.6	61.0
Non-continual			
SUTA	24.0	60.4	53.3
SGEM	25.0	61.0	53.4
Continual			
CSUTA	37.3	83.6	100.3
AWMC	25.8	66.1	60.6
Fast-slow			
DSUTA	24.0	45.6	43.2
w/ Dynamic reset	**22.7**	**39.8**	**35.8**
w/ Fixed reset	22.8	49.4	45.2
w/ Oracle boundary	21.7	36.9	39.5

Table 2: WER (%) of different TTA methods on multi-domain time-varying data. Reported WER is averaged over 3 runs.

overfitting.

4.4.2 Time-varying Multiple Domains

In the following experiment, we set DSUTA with $(M, N) = (5, 5)$ and compare DSUTA with dynamic reset strategy where $(M, N, K, P) = (5, 5, 100, 2)$ on multi-domain time-varying data. We also experiment DSUTA with two baseline reset strategies. 1) ***Oracle boundary*** resets the model at the ground truth domain boundary, and 2) ***Fixed reset*** is the simple fixed-frequency reset strategy, where the reset frequency is set to 50.

Table 2 summarizes the results. DSUTA is comparable to or better than other baseline methods, and applying *Dynamic reset* further boosts the performance. Since we set DSUTA with fewer adaptation steps, our proposed method is both better and faster than SUTA in the multi-domain scenario.

For the non-continual TTA baselines, WER is improved in all cases but remains very high on MD-Hard and MD-Long. For the continual TTA baselines, CSUTA performs worse than the pretrained model due to its instability. For AWMC, the original paper does not test in the multi-domain scenario, and our results show that AWMC is inferior to SUTA in this context.

Regarding the model reset strategy, the proposed *Dynamic reset* outperforms *Fixed reset*. *Fixed reset* performs worse than DSUTA without reset on MD-Hard and MD-Long, suggesting that resetting too frequently might hinder the model from utilizing knowledge from past samples, thereby harming overall performance. Compared to *Oracle boundary* (upper bound), *Dynamic reset* achieves slightly worse performance on MD-Easy and MD-Hard. However, on MD-Long, *Dynamic reset* surprisingly achieves a 35.8% WER, which is even better than the 39.5% WER using *Oracle boundary*. Since *Dynamic reset* automatically determines when to reset, it can further utilize the knowledge from other noise domains when it is beneficial, rather than relying solely on single-domain data for adaptation.

Lastly, DSUTA demonstrates ***superior performance on real multi-domain noisy data***, as shown in Table 1 column "CHiME-3". DSUTA achieves 21.7% WER, while the baseline SUTA and AWMC only yield 23.3% and 22.4% WER, respectively. This result further validates the proposed DSUTA can be generalized to real multi-domain noisy speech data stream.

5 Discussion

5.1 Why Choosing Averaged LII as an Indicator?

A good indicator should *separate in-domain and out-of-domain samples into two clusters*. To visu-

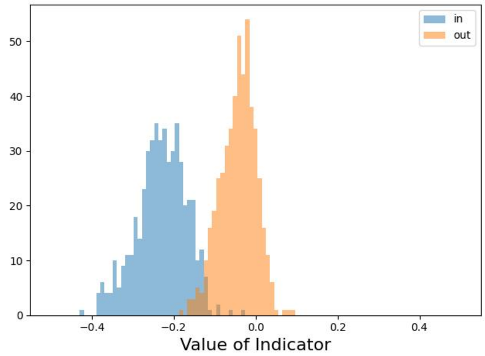

Figure 5: Distributions of averaged LII (over 5 samples) from the GS domain (in) and non-GS domains (out).

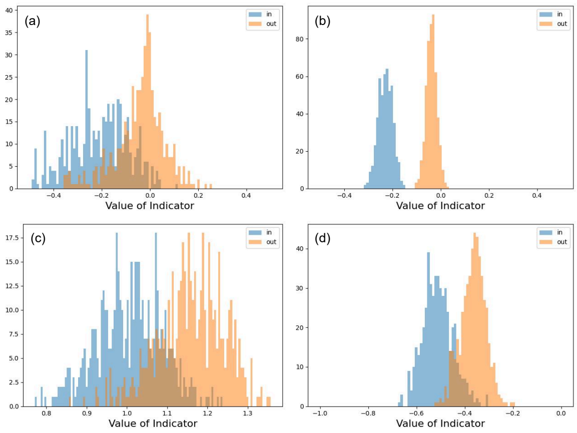

Figure 6: Distributions of other possible indicators. (a): original LII, (b): averaged LII for 20 samples, (c): without subtraction of pre-trained model loss, and (d): with the adapted parameters.

alize the indicator, we selected 500 samples from the GS domain as the source domain and randomly sampled 2000 samples from other domains as out-of-domain samples. $\phi_{\mathcal{D}}$ is then trained on 100 samples from the GS domain using $\mathcal{L}_{suta}$. We randomly sampled 500 averaged LIIs. Figure 5 visualizes the distributions of averaged LIIs (over 5 samples) of the remaining in-domain and out-of-domain samples. By using the averaged LII, two distributions are well separated.

Figure 6 visualizes the distributions of other possible choices of the indicator. Figure 6a, b shows the distribution of averaged LII over 1 sample (i.e., the original LII) and 20 samples, respectively. Using a single sample is not sufficient to distinguish the distributions while considering more samples makes the detection more accurate. Figure 6c illustrates the case without subtracting the loss from the pre-trained model, namely $\mathcal{L}(\phi_{\mathcal{D}}, x_t)$. The distributions are not well separated. In Figure 6d, we also tried using the parameters after adaptation A instead of the meta-parameters, namely

$$\mathcal{L}(A(\phi_{\mathcal{D}}, x_t), x_t) - \mathcal{L}(A(\phi_{pre}, x_t), x_t).$$

However, it resulted in more overlap between the two distributions than the proposed method.

MD-Easy	$s = 20$	$s = 100$	$s = 500$
DSUTA	24.1	23.9	24.0
w/ dynamic reset	**23.8**	23.7	**22.7**
w/ fixed reset	24.6	**23.1**	22.8
w/ oracle boundary	23.7	22.8	21.7
MD-Hard	$s = 20$	$s = 100$	$s = 500$
DSUTA	45.6	44.7	45.6
w/ dynamic reset	**42.3**	**44.5**	**39.8**
w/ fixed reset	53.3	49.9	49.4
w/ oracle boundary	57.3	46.6	36.9

Table 3: WER (%) of different reset strategies on MD-Easy and MD-Hard with different transition rates. Reported WER is averaged over 3 runs. s is the domain transition rate.

5.2 Different Domain Transition Rates

In this section, we investigate *how different domain transition rates affect the performance of reset strategies*. The original transition rate (s) of MD-Easy and MD-Hard is 500. We compare different reset strategies in 3 transition rates: $s = 20, 100, 500$. To maintain a total length of the data stream to 2500, for $s = 100$, the domain order sequence is repeated 5 times, and for $s = 20$, the domain order sequence is repeated 25 times. We follow the hyperparameter settings described in Section 4.4.2.

The results are presented in Table 3. *Oracle Boundary* and *Fixed Reset* show that as the transition rate increases, resetting too often deteriorates performance. This phenomenon is more pronounced in MD-Hard, where DSUTA outperforms SUTA by a large margin, suggesting that continual learning is more effective in this context. *Oracle Boundary* severely deteriorates performance when $s = 20$ and $s = 100$, implying that learning from samples from other noise domains might be beneficial. Since *Dynamic Reset* automatically handles when to reset, it can utilize the knowledge from other noise domains, and reset is not triggered as frequently as in *Oracle Boundary* or *Fixed reset* under fast transitions, leading to better results.

Method	Steps		Runtime (s)	
	#Forward	#Backward	Total	Avg
Non-continual				
SUTA	100000	100000	5040	0.080
SGEM	100000	100000	11620	0.186
Continual				
AWMC	300000	100000	11704	0.187
Fast-slow				
DSUTA	52000	52000	3885	0.062
w/ Dynamic reset	72000	52000	4149	0.066

Table 4: Comparison of Forward/Backward steps and Runtime for different TTA methods on MD-Long. **Avg** is the averaged runtime (s) for a 1-second utterance. The result is averaged over 3 runs.

In summary, the proposed *Dynamic reset* offers good performance across diverse scenarios due to its flexibility. *Dynamic reset* minimizes unnecessary resets and utilizes learned knowledge more effectively, consistently outperforming other reset strategies, making it a versatile solution.

5.3 Efficiency of the Proposed Method

DSUTA is more efficient in adaptation steps than SUTA. Appendix Figure 7 compares SUTA and DSUTA on 10 domains of LS-C under different adaptation steps $N = 0, 1, 3, 5, 10$. DSUTA can use fewer adaptation steps to achieve better performance than SUTA with more adaptation steps.

To assess the efficiency of different TTA methods, we run them on MD-Long and compare the required forward/backward steps and runtime in Table 4. CSUTA is excluded due to its poor performance. We follow the hyperparameter settings described in Section 4.4.2. All experiments were conducted on an Nvidia GeForce RTX 3080Ti GPU. Note that the results are for reference only, as values can slightly differ depending on the implementation. DSUTA is more efficient in the adaptation step and overall faster than SUTA, SGEM, and AWMC. Although adding the dynamic reset strategy slightly increases runtime, it remains faster overall. In conclusion, our method is not only superior in performance but also more efficient than existing approaches.

5.4 Resets Frequency and Occurrence

We propose a dynamic model reset strategy to detect domain shifts, improving both performance and efficiency. However, the frequency of model resets and their positions within the data stream remain unclear. In Table 5, we present the reset timings of our method across three runs. The oracle boundaries occur at steps 500, 1000, 1500, and 2000. The results indicate that the reset timings are close to, but not exactly aligned with, the oracle boundaries.

3 runs	Automatic reset step
MD-Easy-1	540, 1530, 2010
MD-Easy-2	565, 1635, 2025
MD-Easy-3	560, 1530, 2010
MD-Hard-1	155, 555, 790, 1045
MD-Hard-2	510, 1510, 2010
MD-Hard-3	155, 510, 1165, 1510, 2010

Table 5: Reset times and Automatic reset steps for MD-Easy and MD-Hard tasks over 3 runs. The ground truth task boundaries at steps equal to 500, 1000, 1500, and 2000.

6 Conclusion

In this work, we advance the non-continual Test-Time Adaptation (TTA) method for ASR into a continual learning framework using a novel approach to stabilize adaptation and improve performance. Specifically, we introduce Dynamic SUTA (DSUTA), a fast-slow method that combines non-continual and continual TTA, demonstrating significant improvements on single-domain test data. Additionally, we propose a statistical dynamic reset strategy to enhance robustness and performance on time-varying test data streams. Experimental results indicate that our proposed method outperforms the non-continual SUTA baseline and previous continual TTA methods using pseudo labeling.

Limitations

The primary limitations of this paper are as follows:
Domain Shift with Background Noises: In this work, we use noise corruptions to simulate changing domains and control domain shifts. However, there are various other speech domains to study, such as accents, speaker characteristics, and speaking styles. We will consider these domains in future research.
Different Types of End-to-End ASR Models: This work follows SUTA with a CTC-based ASR model, but there are different kinds of end-to-end ASR models available. As shown in (Kim et al., 2023), entropy minimization-based TTA methods can be extended to other end-to-end ASR models. We encourage future research to extend our DSUTA

method to these other end-to-end ASR models.

Not Addressing Model Forgetting: This work focuses on adaptation to testing samples during inference time, rather than memorizing all past knowledge. Consequently, the proposed method might experience catastrophic forgetting as the domain changes. However, given a new test sample, the method can instantly adapt to that instance, ensuring that the final performance remains strong.

Acknowledgments

The authors want to thank the reviewers for their insightful comments and suggestions during the rebuttal period. Guan-Ting Lin is supported by the NTU GICE and the NTU Ph.D. scholarship.

References

Jon Barker, Ricard Marxer, Emmanuel Vincent, and Shinji Watanabe. 2017. The third 'chime'speech separation and recognition challenge: Analysis and outcomes. *Computer Speech & Language*, 46:605–626.

Sachin Goyal, Mingjie Sun, Aditi Raghunathan, and J Zico Kolter. 2022. Test time adaptation via conjugate pseudo-labels. *Advances in Neural Information Processing Systems*, 35:6204–6218.

Changhun Kim, Joonhyung Park, Hajin Shim, and Eunho Yang. 2023. SGEM: Test-Time Adaptation for Automatic Speech Recognition via Sequential-Level Generalized Entropy Minimization. In *Proc. INTERSPEECH 2023*, pages 3367–3371.

Jae-Hong Lee, Do-Hee Kim, and Joon-Hyuk Chang. 2023. Awmc: Online test-time adaptation without mode collapse for continual adaptation. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 1–8. IEEE.

Guan-Ting Lin, Shang-Wen Li, and Hung yi Lee. 2022. Listen, Adapt, Better WER: Source-free Single-utterance Test-time Adaptation for Automatic Speech Recognition. In *Proc. Interspeech 2022*, pages 2198–2202.

Hongfu Liu, Hengguan Huang, and Ye Wang. 2023. Advancing test-time adaptation for acoustic foundation models in open-world shifts. *arXiv preprint arXiv:2310.09505*.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. 2022a. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*, pages 16888–16905. PMLR.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. 2022b. Towards stable test-time adaptation in dynamic wild world. In *The Eleventh International Conference on Learning Representations*.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210.

Ori Press, Steffen Schneider, Matthias Kümmerer, and Matthias Bethge. 2024. Rdumb: A simple approach that questions our progress in continual test-time adaptation. *Advances in Neural Information Processing Systems*, 36.

Chandan KA Reddy, Ebrahim Beyrami, Jamie Pool, Ross Cutler, Sriram Srinivasan, and Johannes Gehrke. 2019. A scalable noisy speech dataset and online subjective test framework. *Proc. Interspeech 2019*, pages 1816–1820.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. 2020. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. 2022. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211.

A Appendix

A.1 Different noise levels

From Table 1 and Table 2, we observe a trend that DSUTA has a larger advantage over other methods under severe domain shift where the pre-trained model performs poorly. To investigate *how different levels of domain shift affect the proposed method*, we compare the pre-trained model, SUTA, and DSUTA with noise levels of 0dB, 5dB, and 10dB on the AC, SD, and TP domains from LS-C, which are the top 3 well-performing domains for the pre-trained model. We set $N = 5$ for both SUTA and DSUTA. Table 6 summarizes the results.

The results show that DSUTA is more effective under severe corruption. As the noise level decreases, although DSUTA outperforms the pre-trained model, SUTA becomes better than DSUTA. We hypothesize that while DSUTA is quite effective in noisy speech, its performance gain over the non-continual version (SUTA) is limited to relatively clean speech. Improving DSUTA's performance over SUTA on clean speech remains an area for future work.

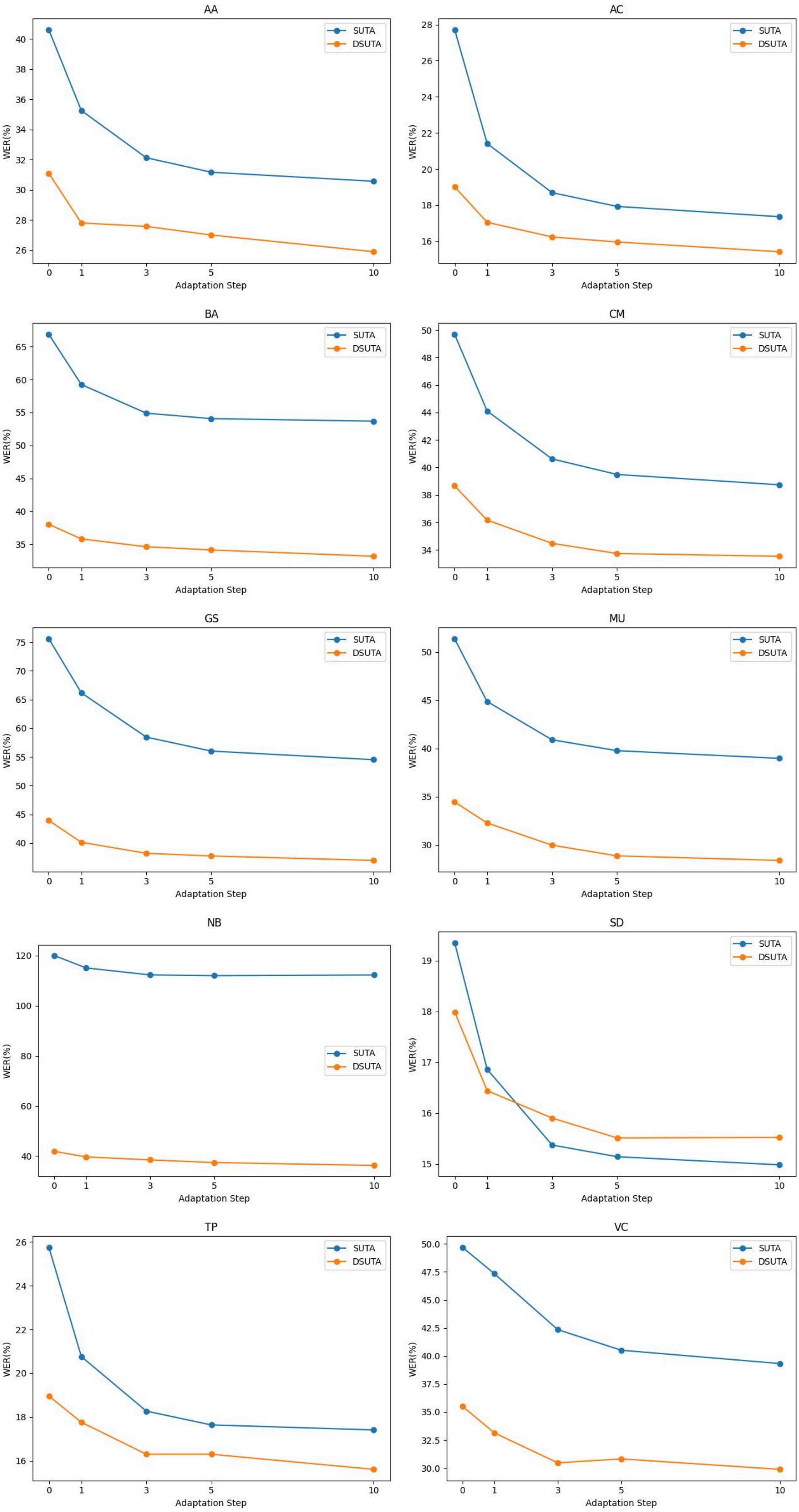

Figure 7: WER (%) of different number of adaptation steps on 10 noise domains of LS-C.

Domain	**Method**	**0dB**	**5dB**	**10dB**
AC	Pre-trained	63.7	27.7	14.2
	SUTA	39.5	17.4	**10.6**
	DSUTA	**27.6**	**16.0**	11.5
SD	Pre-trained	29.7	19.4	13.6
	SUTA	23.6	**15.0**	**10.8**
	DSUTA	**22.4**	15.5	12.0
TP	Pre-trained	42.4	25.8	16.6
	SUTA	28.8	17.4	**12.1**
	DSUTA	**22.4**	**16.3**	12.4

Table 6: WER(%) comparison for different noise levels. Reported WER is averaged over 3 runs.

A.2 Hyper-parameter Tuning

We explore different hyper-parameters for DSUTA with the dynamic reset strategy. We use MD-Long as the data sequence. Table 7 presents the results for various buffer sizes M. Our proposed method performs well overall. A smaller buffer size can make the update of meta-parameters unstable, while a larger buffer increases latency in triggering model reset after a domain shift since the shift is detected once every M steps. Therefore, a medium buffer size is preferred.

Table 7 also presents the results for different K values during the domain construction stage. Again, our proposed method performs well overall. The performance of $K = 50$ is worse than $K = 100$ and $K = 200$, suggesting that domain construction benefits from having enough steps to collect LII statistics and train a domain-specialized model ϕ_D.

A.3 Generalization to Different Source ASR Models

To test the generalization of the proposed method, we adopt other source ASR models with DSUTA and dynamic reset strategy. Table 8 reports the results with the ASR model fine-tuned from wav2vec 2.0-base, data2vec-base[8], and HuBERT-large[9] model. All the ASR models are trained with Librispeech 960 hours. Results show that both DSUTA and DSUTA with the dynamic reset strategy perform effectively across different models, yielding significantly better WER than the pre-trained model and the SUTA.

Setup	**WER**
$M = 3$	36.8
$M = 5$	**35.8**
$M = 10$	37.0
$K = 50$	38.5
$K = 100$	35.8
$K = 200$	**35.5**

Table 7: WER(%) comparison of different hyperparameters on MD-Long. Reported WER is averaged over 3 runs.

Method	wav2vec2-base	data2vec-base	hubert-large
Pre-trained	61.0	59.6	43.3
SUTA	53.3	53.3	39.3
DSUTA	43.2	52.0	**17.8**
w/ Dynamic reset	**35.8**	**46.3**	19.0

Table 8: WER(%) comparison of different CTC-based ASR models on MD-Long. Reported WER is averaged over 3 runs.

A.4 Objective of SUTA (L_{suta})

Assume C is the number of output classes and L is the number of frames in the utterance. $\mathbf{P}_{\cdot\mathbf{j}} \in \mathbb{R}^L$ denotes the output probabilities of the j-th class of the L frames.

Entropy Minimization (EM):

$$\mathcal{L}_{em} = \frac{1}{L}\sum_{i=1}^{L}\mathcal{H}_i = -\frac{1}{L}\sum_{i=1}^{L}\sum_{j=1}^{C}\mathbf{P_{ij}}\log\mathbf{P_{ij}}.$$

Minimum Class Confusion (MCC):

$$\mathcal{L}_{mcc} = \sum_{j=1}^{C}\sum_{j'\neq j}^{C}\mathbf{P}_{\cdot\mathbf{j}}^{\top}\mathbf{P}_{\cdot\mathbf{j'}}.$$

The final SUTA objective is defined as a mixture of $\mathcal{L}_{em}$ and $\mathcal{L}_{mcc}$:

$$\mathcal{L}_{suta} = \alpha\mathcal{L}_{em} + (1-\alpha)\mathcal{L}_{mcc}.$$

We follow the settings in the original paper, which set $\alpha = 0.3$ and apply temperature smoothing on logits with a temperature of 2.5.

[8] https://huggingface.co/facebook/data2vec-audio-base-960h

[9] https://huggingface.co/facebook/hubert-large-ls960-ft

Algorithm 2 Dynamic SUTA with the dynamic reset strategy

```
Input: Data Sequence $\{x_t\}_{t=1}^T$, buffer $\mathcal{B}$ with size $M$, adaptation step $N$, number of samples for
    construction $K$, patience $P$, pre-trained parameters $\phi_{pre}$
Output: Predictions $\{\widehat{y}_t\}_{t=1}^T$
$\mathcal{B}, \phi_1 \leftarrow \{\}, \phi_{pre}$
$k, last\_reset, stats \leftarrow \lfloor K/2 \rfloor, 0, \{\}$
$Results \leftarrow \{\}$
for $t = 1$ to $T$ do
    $\widehat{\phi}_t \leftarrow \phi_t$                                        ▷ SUTA as adapt algorithm
    for $n = 1$ to $N$ do
        $\mathcal{L} \leftarrow \mathcal{L}_{suta}(\widehat{\phi}_t, x)$
        $\widehat{\phi}_t \leftarrow$ Optimizer$(\widehat{\phi}_t, \mathcal{L})$
    $\widehat{y}_t \leftarrow \widehat{\phi}_t(x_t)$                            ▷ Inference and save the prediction
    $Results \leftarrow Results \cup \{\widehat{y}_t\}$
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{x_t\}$
    if $t\%M = 0$ then                                   ▷ Update meta-parameter every $M$ steps
        if $t > last\_reset + K$ and IsReset$(\mathcal{G}, \mathcal{B}, P)$ then              ▷ Dynamic reset
            $\phi_{t+1} \leftarrow \phi_{pre}$
            $last\_reset \leftarrow t$
        else
            $\mathcal{L} \leftarrow \frac{1}{M} \sum_{x \in \mathcal{B}} \mathcal{L}_{suta}(\phi_t, x)$
            $\phi_{t+1} \leftarrow$ Optimizer$(\phi_t, \mathcal{L})$
        $\mathcal{B} \leftarrow \{\}$
    else
        $\phi_{t+1} \leftarrow \phi_t$
    if $t = last\_reset + k$ then                        ▷ Save the domain-specialized model
        $\phi_{\mathcal{D}} \leftarrow \phi_t$
    else if $last\_reset + k < t \leq last\_reset + K$ then                ▷ Collect LII stats
        $stats \leftarrow stats \cup \{LII_t\}$
    if $t = last\_reset + K$ then                              ▷ Generate distribution
        $\mathcal{G} \leftarrow \mathcal{N}(\mu_{stats}, \sigma^2_{stats})$
return $Results$
```