Exploiting the Availability-Continuity Trade-off in Imperfect Retraining of Machine Learning Systems

Zhengji Wang and Fumio Machida
Department of Computer Science
University of Tsukuba
Tsukuba, Japan
wang.zhengji@sd.cs.tsukuba.ac.jp and machida@cs.tsukuba.ac.jp

***Abstract*—Machine Learning Systems (MLSs) often combine diverse models to achieve complex objectives but face performance degradation due to dataset shifts. Regular performance monitoring and model retraining are essential to mitigate this risk. However, model retraining may not always fully restore the system's performance, which is known as the imperfect retraining problem. This study examines model retraining policies to maintain MLS performance in the face of imperfect retraining. First, we demonstrate real-world applications that encounter imperfect retraining in computer vision and natural language processing tasks. Next, we theoretically analyze two retraining policies, progressive and conservative, to counteract performance degradation. We formulate the dynamics of model degradation and retraining using semi-Markov processes and quantitatively evaluate service availability and continuity, which measures how long the service can maintain its performance. The numerical analysis results demystify the notable trade-off between service availability and continuity, guiding a proposed retraining strategy to better sustain MLS performance.**

***Index Terms*—semi-Markov process, dataset shift, machine learning system, retraining, service availability**

I. Introduction

Machine Learning (ML) has progressed dramatically over the past two decades, from laboratory curiosity to a practical technology in widespread commercial use [2]. With the fast development of both ML algorithms and computing hardware, ML models have been applied in many application domains such as medical and commercial ads, performing tasks such as classification and clustering [4]. Machine Learning System (MLS) is defined as a system that incorporates different ML models as its components. Compared to a simple ML model, MLS provides a divide-and-conquer approach to handling complex tasks, such as image captioning, search advertising, and personalized recommendations [5] [6] [7].

Despite its wide applicability, the MLS constantly faces the threat of dataset shift. Dataset shift occurs when the testing data undergoes a phenomenon that alters the distribution of features or the boundary between classes [8]. As a result, the common assumption that the training and testing data follow the same distributions is often violated in real-world applications, leading to performance degradation of the component model. To restore performance, retraining component models periodically is necessary [9]. However, some retraining effort might not benefit the system overall, which is defined as imperfect retraining [1]. This poses a threat to the operations of the MLS. Understanding how to mitigate the impact of imperfect retraining becomes crucial to maintaining the performance of MLS. There are very few studies that mention such a risk, and no systematic solution has ever been proposed [1] [3].

Our study focuses on investigating the impact of the entangled enhancement, a specific type of imperfect retraining, on an MLS with two components. The entangled enhancement is the imperfect retraining caused by component entanglement in the MLS. The impact of entangled enhancement of MLS has been investigated theoretically using Continuous Time Markov Chain (CTMC) [1]. Although the study pioneered the analysis of imperfect retraining, it has several limitations. First, the phenomenon of imperfect retraining is argued on the assumptions and lacks real-world examples. Second, the assumption of CTMC implies that the recovery of the component performance following an exponential distribution might not hold in reality. Thus, the model may not faithfully track the actual dynamics of the retraining process. Moreover, the existing study examines the performance of MLS only in terms of service availability. However, frequent retraining may harm the continuity due to service interruptions [46]. The maintenance policies should also be evaluated in light of this continuity aspect.

To overcome these limitations, we first highlight the real-world problem of entangled enhancement in computer vision and natural language processing tasks. In both applications, we encountered imperfect retraining problems that may degrade the overall system performance after component model retraining. Next, we leverage a semi-Markov process (SMP) to model the dynamics of MLSs confronting dataset shift and imperfect retraining. Unlike CTMC, the transition times are not restricted to the exponential distribution. Thus, we can conduct more comprehensive evaluations of different maintenance policies. Moreover, we propose a new metric, service continuity, that quantifies the expected time to maintain the required service quality without interruptions. Through numerical analysis of the proposed SMP, we find that a notable trade-off between availability and continuity is significantly influenced by the chosen maintenance policies. During the retraining process of the entangled MLS, depending on the specific use case, the trade-off relationship can be leveraged by selecting the appropriate retraining policies to achieve higher service availability or continuity in performing the job.

To sum up, the paper makes the following contributions:

- We showcase the real problem of entangled enhancement through the experiments of a 3D object detection system and a sentence classification system.
- We model the dynamics of the MLS with imperfect retraining under different maintenance policies using SMP.
- We propose a new metric, service continuity, which enables the evaluation of the stability aspect of the MLS.
- We show the notable trade-off between service availability and service continuity, and a strategy guide to choose the effective maintenance policy.

The rest of this paper is organized as follows. Section II presents the related work. Section III explains the problem setting in detail. Section IV showcases the examples of imperfect retraining in different scenarios. Section V proposes the state-space models of the MLS maintenance using SMP. Section VI shows our findings from the numerical analysis results. Section VII discusses the strategy to choose retraining policies. Section VIII explains the limitations of the work. Finally, Section IX concludes the paper.

II. Related Work

A. Dataset Shift

Dataset shift is the phenomenon in which training and test data follow distributions that are in some way different [8]. Dataset shifts have posed threats to ML models by negatively impacting the performance of models trained from past data when applied to new data [12] [13]. To counter dataset shift, certain supervision for the model performance, as well as necessary operational measures, is needed. For example, several approaches using unsupervised techniques to monitor data against the dataset shift and retraining the model from time to time have been proposed [14] [18]. However, the risk of such an approach causing imperfect retraining in the MLS was overlooked. Most of the works discussing methods to counter dataset shift limit the scope of the problem to a single ML model component without considering the context of MLS.

B. Operations in Machine Learning System

Despite the rapid advancement of ML in recent years, many ML proofs of concept have not progressed to production due to insufficient emphasis on operational techniques [19]. Manual operation of ML models is challenging due to the complexity of software and hardware components, necessitating robust automation [21]. Additionally, models must be retrained on new data due to dataset shifts, further emphasizing the need for automation [29]. Machine Learning Operations (MLOps) has emerged as a developing paradigm in recent years, aiming to address these needs by ensuring the reliable deployment and maintenance of models [30]. However, automation in MLSs faces challenges due to component entanglement. For instance, performance degradation after an update makes it difficult to identify the responsible component [31]. Furthermore, self-defeating improvements, where upstream model enhancements may not benefit or may even degrade downstream models, contrast with traditional software systems [3] [32]. Despite extensive discussion of these issues, existing literature lacks approaches to mitigate them within the MLOps framework.

C. Availability Analysis of System

Availability is one metric that reflects the dependability of the system, which is defined as the ability to initiate a service request when desired [34]. To investigate the service availability of a specific system, it is common to use a conceptual model based on a stochastic process for further evaluation. For example, a continuous-time Markov chain (CTMC) has been used to model telecommunications switching systems and computer systems [35] [36]. More flexible approach by using SMP modeling method with different assumptions on the state sojourn time has also been applied to different scenarios such as the software aging problem, reliability of mechanical manufacturing, as well as power supply availability [37] [38] [39]. To the best of our knowledge, our work is the first to apply SMP to model an MLS with model retraining policies.

III. Problem Setting

Imperfect retraining in an MLS is defined as the retraining attempt that does not benefit the overall system performance when countering against the dataset shift [1]. The phenomenon of imperfect retraining can be attributed to a poor selection of training samples [8] or overfitting the algorithm [44], thereby causing the performance deterioration of the component, which fails to benefit the system output. In contrast to these failures in training, rather than considering such a scenario, we focused on a specific type of imperfect retraining, namely entangled enhancement. Entangled enhancement refers to retraining attempts that improve the component model's performance, but due to component entanglement, the improvement fails to benefit the system's output.

Let us consider a sequential MLS consisting of two ML models u and d, where u represents the upstream model and d represents the downstream model. The upstream model u processes part of the input, and its output, combined with another input subset, is passed to the downstream model d to produce the system's final output. The system's performance is measured using a real-world dataset X_{real} against a predefined metric threshold (e.g., Precision or Recall). Initially, both models perform well and meet the threshold, ensuring the system is available, as shown in Figure 1. However, as X_{real} evolves due to dataset shifts, the model's performance can degrade.

To counter such an effect, dataset shifts are monitored, and retraining is triggered when a shift is detected. Each model uses a fixed algorithm, training data, and hypothesis set during retraining. Importantly, retraining d does not affect u, and the system's availability ultimately depends on whether d meets the performance threshold.

In our formulated system, which has two ML models as components, the entangled enhancement is triggered particularly when the upstream model u does not reach the threshold

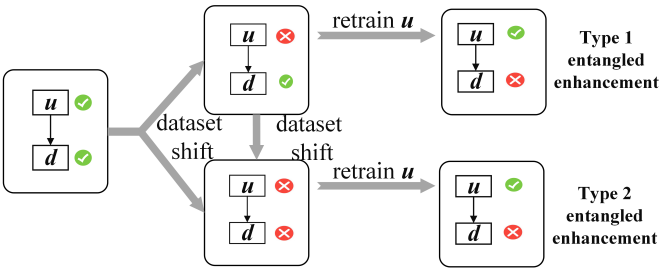


Fig. 1: Type 1 and Type 2 component entangled enhancement in problem setting

due to dataset shift. We classify the dataset shifts into two types, which are illustrated in Figure 1:

1) Type 1: When the upstream model u does not meet the threshold due to dataset shift, but the downstream model d does, a retraining attempt on the upstream model could potentially degrade the performance of the downstream model.
2) Type 2: When both the upstream and downstream models fail to meet their thresholds due to a dataset shift, a retraining attempt on the upstream model results in an entangled enhancement, leading to the downstream model performance still not being recovered.

We presented two experiments in Section IV, demonstrating that retraining the upstream model, which directly improves its performance, actually has a detrimental effect on the downstream model.

IV. Experiments on Imperfect Retraining

To demonstrate real examples of imperfect retraining, we conduct two experiments of entangled enhancement in the tasks of computer vision and natural language processing, respectively [1]

A. Pseudo-Lidar System for 3D Object Detection in Autonomous Driving

In this experiment, we recreated the phenomenon of imperfect retraining, specifically component entanglement, in a 3D object detection system for an autonomous driving scenario, as also considered in [3]. Fig. 2 shows a widely applied MLS in the self-driving car setting. The data input is two images shot from the left and right cameras. The system utilizes a pseudo-LiDAR [20] as the basis for 3D detection of cars. For the upstream model u, PSMNet is adopted [15] to predict the disparity between these two models. The prediction result, together with the original 2D images, would be processed to generate 3D point clouds. For the downstream model, Point R CNN is adopted [16]. The Point R CNN model takes the 3D point clouds as input and generates the car's location in 3D space as output. The dataset for the experiment comes from the Kitti Dataset [17]. While the original study in [3] used AP_{BBOX} and AP_{3D} as their chosen metrics, and also conducted experiments on pedestrians, our experiments differ from the previous study as we limit our scenario to a two-model system, thus only paying attention to car's detection result. Also, we selected AP_R40, which uses 40 recall positions instead of the 11 recall positions proposed in the original Pascal VOC benchmark [11], thus making this metric a fairer comparison of the results [22]. We also include bird's eye view (BEV) and average orientation similarity (AOS), thus incorporating all possible metrics according to the Kitti Dataset standard to make the result more inclusive.

[1]Code and data are available at our GitHub repository.

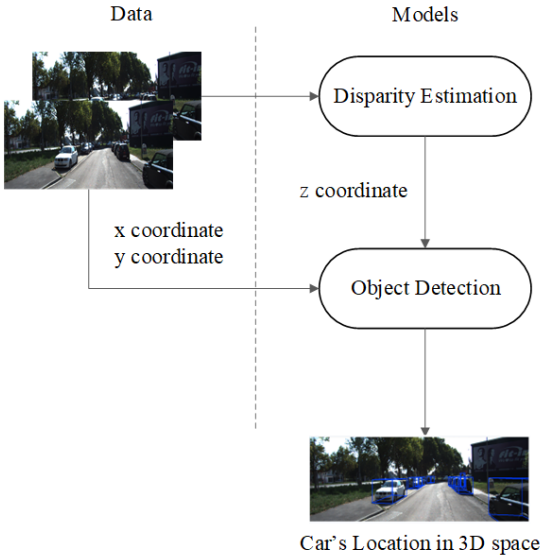


Fig. 2: The MLS for the experiment regarding computer vision. The data consists of two images, one taken by the left camera and the other by the right camera. The upstream model, PSMNet, uses the images to predict the depth information. The downstream model Point RCNN utilizes the depth information provided by the upstream model, along with the x and y coordinates from 2D images, to predict the car's location in 3D space.

a) Upstream Model: We trained PSMNet with two training loss functions: (1) the depth mean absolute error and (2) the disparity mean absolute error (MAE). Depth is proportional to the reciprocal of the disparity. We assume that the PSMNet trained using depth MAE is the original upstream model, and is denoted as u, and the PSMNet trained using disparity MAE is the retrained model, and is denoted as u'. After training these two versions of models using different loss functions, the MAE of each model for disparity on the validation set is shown in TABLE I.

It can be observed that, indeed, the upstream model's prediction ability increases across the entire range, as the MAE of the disparity decreases from 1.28 to 1.21. The improvement effect is dramatic for close-range objects within 10 meters, in which the MAE of the disparity is reduced from 1.61 to 1.38. However, for objects that are farther from the camera, such as those within a range of 10 meters to 20 meters, the disparity estimation result actually has a slightly higher MAE,

TABLE I: The disparity's MAE before and after the update

PSMNet		
range	original model u	retrained model u'
0-10m	1.61	**1.38**
10-20m	**1.0**	1.01
over 20m	**1.42**	1.50
Whole range	1.28	**1.21**

increasing from 1.0 to 1.01. Specifically for objects over 20 meters, the MAE increases from 1.42 to 1.50, indicating a decrease in estimation accuracy.

b) Downstream Model: By using the estimations from u and u', we process two versions of the pseudo-LiDAR point clouds as inputs to train two versions of the downstream model, which are both Point R CNNs. Besides the input data, there is no change in the training algorithm. The result is shown in TABLE II:

TABLE II: AP_R40 of detection result at IoU threshold of 0.7

Point R CNN		
metrics	original model u	retrained model u'
3D	**44.75**	42.62
BEV	**56.02**	54.06
AOS	**76.44**	74.15
BBOX	**79.17**	76.36

The evaluation metric employed for the downstream model is AP_R40 with an Intersection over Union (IoU) threshold of 0.7. The common practice of setting the IoU threshold to 0.7 aligns with the Kitti Benchmark and is consistent with prevailing standards in diverse object detection literature [16] [20]. In contrast to the traditional AP, which utilizes 11 recall positions as per the original Pascal VOC benchmark, AP_R40 encompasses 40 recall positions, rendering it a more comprehensive measure for assessing object detection outcomes [22]. Four distinct metrics, namely AP_R40_{3D}, AP_R40_{BEV}, AP_R40_{AOS}, and AP_R40_{BBOX}, are employed to evaluate various aspects of detection performance, representing 3D bounding box accuracy, 2D bird's-eye view accuracy, average orientation similarity, and 2D bounding box accuracy, respectively.

Notably, an assessment of the downstream model based on both AP_R40_{3D} and AP_R40_{BEV} at an IoU threshold of 0.7 reveals that, despite an enhancement in the upstream model's performance, the resultant output leads to diminished accuracy in the downstream model's ability to detect the location of cars. Specifically, the downstream model's performance declines from 44.75 to 42.62 for AP_R40_{3D} and from 56.02 to 54.06 for AP_R40_{BEV}. Similar deterioration is observed for AP_R40_{AOS} and AP_R40_{BBox}, which decrease from 76.44 to 74.15 and from 79.17 to 76.36, respectively.

This decline can be attributed to the upstream model's heightened proficiency in estimating disparities for short-range objects, accompanied by a compromise in its ability to accurately estimate disparities for objects at greater distances. Given that cars are typically positioned farther from the camera, they fail to benefit proportionally from the improvement, resulting in a paradoxical performance degradation.

c) Result Analysis: The experiment replicates the complete process of updating an upstream model. The result confirms that the phenomenon of component entanglement indeed can happen in a two-component MLS when the upstream model is retrained. In the proposed system, we retrained the upstream model of disparity estimation and reduced its disparity MAE on the validation set. However, the downstream model for detecting cars actually shows a decrease in precision, thus showcasing the validity of imperfect retraining, which is also the entangled enhancement in a two-component MLS.

B. Sentence Classification System for Social Media

In this experiment, we created the phenomenon of component entanglement in a sentence classification system for information collected from social media. For the upstream model u, a feature extraction model is adopted to extract the numerical vector from the sentence. We use BERT and TextCNN as the different versions of the upstream model before and after the retraining [23] [24]. The extracted features, concatenated with the raw features extracted from the sentence using tf-idf, as well as other statistical features such as the number of nouns in the sentence, would be used as the input for the downstream model. For the downstream model d, XGBoost is adopted [25]. XGBoost utilizes the concatenated features to generate the final classification of the input sentence. It is worth noting that the upstream feature extraction model is also trained for the sentence classification task. The reason why upstream model is not directly used for our final task but is used as feature extraction model is because of the following reasons: First, the use of BERT and TextCNN to extract features allows the fusion with other features such as the vector processed by tf-idf or word2vec, thus making a better prediction result possible [26]. Second, the upstream model's output can be reused for other downstream tasks, such as hate-speech detection or general sentiment analysis [27] [28]. We collected 354211 sentences under different sections of topics from a social media platform, Weibo, as our dataset, in which each sentence belongs to one of the following classes: Market Monitoring, Finance Industry, Social Welfare, Education Industry, Food Security, Public Security, Crime Case, and Medical Health.

a) Upstream Model: We trained BERT and TextCNN by viewing them as sentence classification models and attached a softmax function to transform the features into probabilities. The loss function we used is multi-class cross-entropy. We assume that the upstream model trained using TextCNN is denoted as u, and the upstream model trained using BERT is denoted as u'. After training these two versions of models for the classification task, the accuracy of each model on the validation set is shown in TABLE III. It can be observed that, compared to the TextCNN model, the retrained upstream BERT model indeed improves overall performance, as the classification accuracy increases from 0.869 to 0.873.

TABLE III: The feature extraction quality as a classification task before and after the update

TextCNN / BERT		
Class	original model u	retrained model u'
Market Monitoring	**0.870**	0.865
Finance Industry	0.872	**0.876**
Social Welfare	0.861	**0.867**
Education Industry	**0.884**	0.868
Food Security	0.856	**0.868**
Public Security	0.874	**0.899**
Crime Case	0.864	**0.872**
Medical Health	0.876	**0.880**
All Classes	0.869	**0.873**

b) Downstream Model: We removed the softmax layer for classification from the upstream TextCNN model u and the BERT model u', and utilized the remaining final layer as the extracted feature. By concatenating the same features directly processed from the original data using tf-idf and other statistical information, such as sentence norms, we processed two versions of the extracted features from the sentences. We then use the features to train two versions of the downstream models, both of which are XGBoost models. Besides the input data, there is no change in the training algorithm. The result is shown in Table. IV.

TABLE IV: The accuracy of classification before and after the update

XGBoost		
Class	original model u	retrained model u'
Market Monitoring	0.876	**0.887**
Finance Industry	0.917	**0.919**
Social Welfare	**0.908**	0.899
Education Industry	0.960	**0.968**
Food Security	0.899	**0.907**
Public Security	0.922	**0.938**
Crime Case	**0.917**	0.860
Medical Health	**0.972**	0.971
All Classes	**0.924**	0.921

It is shown that component entanglement improvement indeed occurs, as the downstream model using the features extracted by the original model u of TextCNN has an accuracy of 0.924, while using the updated model u' of BERT results in a decrease in accuracy to 0.921. After retraining, the upstream model provides extracted features that are supposedly more effective in a classification task. However, after the feature fusion and the XGBoost process, the downstream model's result deteriorates.

c) Result Analysis: The experiment replicates the complete process of updating an upstream model in a sentence classification setting. The result again confirms that the phenomenon of component entanglement can indeed occur in a two-component MLS when the upstream model is retrained. In the proposed system, we retrained the upstream model of feature extraction, and its classification accuracy was increased on the validation set. However, the downstream model for the actual classification shows a decrease in accuracy, thus showcasing the threat of entangled enhancement in the context of the sentence classification task.

V. System Maintenance Modeling

A. Motivation

As shown in Section IV, MLS in different application scenarios faces the threat of entangled enhancement. This phenomenon occurs when retraining appears to improve the performance of a component but ultimately worsens the overall system performance. One of the practical solutions to mitigate the issue of entangled enhancement is the maintenance policy that guides every retraining attempt, such as when to retrain upstream or downstream models. By tailoring retraining policies to the specific system, it is possible to maximize service availability [1]. However, different retraining policies affect not only service availability but also other system aspects as well. Frequent retraining can improve service availability, but it may also disrupt service continuity due to interruptions. A higher availability can be achieved at the cost of service continuity. Thus, we need to investigate the potential trade-off between availability and continuity of MLS.

To analyze service availability and continuity under different maintenance policies, we can resort to state-space modeling and analysis, as adopted in [1]. However, modeling using CTMC, which assumes the Markov property throughout the retraining and failure process, is overly restrictive. It is unlikely that all the retraining times follow the exponential distributions. To overcome the limitation, we leverage SMP to model the state transitions of the MLS. In the following sections, we first provide the definition of SMP for the two-component MLS and propose SMP models for two maintenance policies: progressive retraining and conservative retraining. Progressive retraining continuously attempts to retrain the model proactively, mitigating the risk of performance degradation due to dataset shift. Conservative retraining, on the other hand, attempts model retraining only when the system manager identifies the degraded performance.

B. Definition

We model the MLS by an SMP $\{X(t) \mid t \geq 0\}$. In reference to the performance thresholds, we define the state of the two-component MLS as a pair $(s(u, \tau_u), s(d, \tau_d))$ where

$$s(x, \tau_x) = \begin{cases} 1, & \text{if model } x \text{ satisfies threshold } \tau_x, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

We assume that the initial state of the system is $(0, 0)$, representing that both upstream model u and downstream model d meet the performance thresholds (i.e., τ_u and τ_d, respectively). Additionally, we denote $(0, 0')$ and $(0, 1')$ as vanishing states, representing that the states $(0, 0)$ and $(0, 1)$ are altered due to retraining and are about to transition to another state immediately. We use the kernel function to define our SMP,

$$k_{ij}(t) = P\left\{X_{n+1} = j, T_{n+1} - T_n \leq t \mid X_n = i\right\}, \quad (2)$$

as the conditional probability that, given that the process has entered state i at time T_n, the next transition occurs at time t toward state j. We collect all the $k_{ij}(t)$ into a matrix, called the kernel matrix of SMP:

$$\boldsymbol{K}(t) = [k_{ij}(t)]. \tag{3}$$

We assume that the embedded discrete-time Markov chain (DTMC) of the SMP satisfies the homogeneity property and hence it is fully characterized by its one-step transition probability matrix (TPM) $\boldsymbol{P} = [p_{ij}]$, where p_{ij} is the probability that the next jump of the DTMC is to state j given that the process is in state i. We define $H_i(t)$ to be the sojourn time distribution in state i. Specifically,

$$p_{ij} = \lim_{t\to\infty} k_{ij}(t), \quad \boldsymbol{P} = \lim_{t\to\infty} \boldsymbol{K}(t), \quad H_i(t) = \sum_j k_{ij}(t). \tag{4}$$

C. Progressive Retraining Model

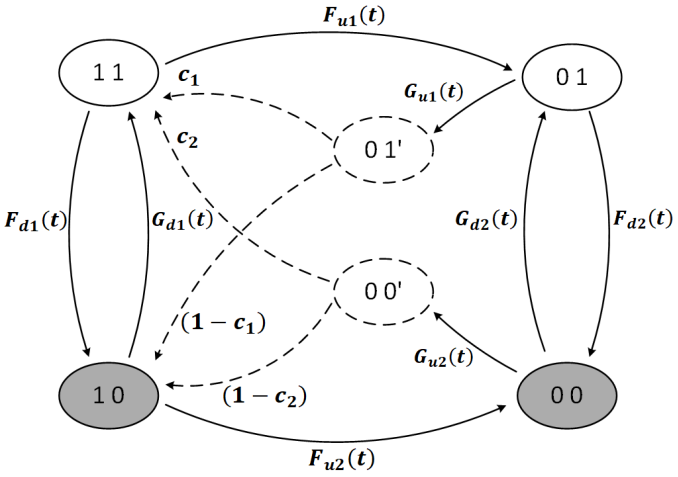


Fig. 3: A SMP depicting the dynamics of MLS under the progressive retraining policy. The sojourn time of the transitions caused by dataset shift is set to follow general functions $F_{u1}(t)$, $F_{u2}(t)$, $F_{d1}(t)$, and $F_{d2}(t)$. The sojourn time of the transitions caused by retraining is set to follow some general distribution function. $G_{u1}(t)$ denotes the sojourn time distribution from $(0,1)$ to $(0,1')$. $G_{u2}(t)$ denotes the sojourn time distribution from $(0,0)$ to $(0,0')$. $G_{d1}(t)$ denotes the sojourn time distribution from $(1,0)$ to $(1,1)$. $G_{d2}(t)$ denotes the sojourn time distribution from $(0,0)$ to $(0,1)$. $(0,1')$ and $(0,0')$ are vanishing states with sojourn time of 0. c_1 denotes the coverage factor that once $(0,1')$ is entered, it will be transited to $(1,1)$. c_2 denotes the coverage factor that once $(0,0')$ is entered, it will be transited to $(1,1)$.

We model the state transitions of a two-component MLS under the progressive retraining policy by the SMP, as shown in Figure 3. We denote the the sojourn time of state transitions caused by dataset shift, which are $(1,1) \to (0,1)$, $(1,0) \to (0,0)$, $(1,1) \to (1,0)$ and $(0,1) \to (0,0)$, to follow some general distribution $F_{u1}(t)$, $F_{u2}(t)$, $F_{d1}(t)$ and $F_{d2}(t)$ respectively. We denote the the sojourn time of state transitions caused by retraining, which are $(0,1) \to (0,1')$, $(0,0) \to (0,0')$, $(1,0) \to (1,1)$ and $(0,0) \to (1,1)$, to follow some general distribution $G_{u1}(t)$, $G_{u2}(t)$, $G_{d1}(t)$ and $G_{d2}(t)$ respectively. Vanishing states $(0,1')$ and $(0,0')$ have a sojourn time of 0. c_1 is the coverage factor for type 1 entangled enhancement. c_2 is the coverage factor for type 2 entangled enhancement.

At state (1,1), no recovery should be observed. Due to the dataset shift, the system might experience a compromise in either the upstream model u or the downstream model d. At state (0,1), the upstream model u fails to reach the threshold while the downstream model d remains above the threshold. Therefore, only the successful retraining to the upstream model u which leads to state $(0,1')$ can be observed. If the retraining is successful, it leads to a transition from state $(0,1')$ to state $(1,1)$, or it could trigger a Type 1 entangled enhancement, which transits from state $(0,1')$ to state $(1,0)$. At state $(0,0)$, both upstream and downstream models fail to reach the threshold, and hence go under retraining attempts, and attempts that change the state of SMP would lead to a transition to $(0,0')$. However, retraining the upstream model can have two consequences: 1. the upstream model recovers the performance, while the downstream model fails to recover the performance τ_d. This could be caused by either a Type 2 entangled enhancement, or the downstream model only is improved slightly, which is not enough for the state change. This would make SMP transits to $(1,0)$. 2. The retraining is successful enough that the downstream model also benefits from the improvement, so that it reaches the threshold τ_d. This would make SMP transits to $(1,1)$. At state $(1,0)$, only retraining the downstream model can restore it to $(1,1)$.

In order to derive the steady-state probability of the SMP, we need to obtain the steady-state probability vector $\boldsymbol{v}$ of the embedded DTMC by solving the linear system of equations $\boldsymbol{v} = \boldsymbol{v}\boldsymbol{P}$ w.r.t $\boldsymbol{v}\boldsymbol{e}^{\mathrm{T}} = 1$, where e is the vector with all entries equal to 1. In SMP, TPM can be deduced by using the Equation 4, therefore, we can obtain the following $\boldsymbol{P}$ matrix:

$$\boldsymbol{P} = \begin{array}{c} \begin{matrix} (1,1) & (0,1) & (1,0) & (0,0) & (0,1') & (0,0') \end{matrix} \\ \begin{pmatrix} 0 & P_1 & 1-P_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-P_2 & P_2 & 0 \\ P_3 & 0 & 0 & 1-P_3 & 0 & 0 \\ 0 & P_4 & 0 & 0 & 0 & 1-P_4 \\ c_1 & 0 & 1-c_1 & 0 & 0 & 0 \\ c_2 & 0 & 1-c_2 & 0 & 0 & 0 \end{pmatrix} \end{array}, \tag{5}$$

$$\begin{cases} P_1 = \int_0^\infty (1 - F_{d1}(t)) dF_{u1}(t), \\ P_2 = \int_0^\infty (1 - F_{d2}(t)) dG_{u1}(t), \\ P_3 = \int_0^\infty (1 - F_{u2}(t)) dG_{d1}(t), \\ P_4 = \int_0^\infty (1 - G_{u2}(t)) dG_{d2}(t). \end{cases} \tag{6}$$

The steady-state probability $\boldsymbol{v}$ of the embedded DTMC can be derived. Let $h_i = \int_0^\infty (1 - H_i(t))dt$ be the mean sojourn time in state i, then we have:

$$\begin{cases} h_{(1,1)} = \int_0^\infty (1 - F_{u1}(t))(1 - F_{d1}(t))dt, \\ h_{(0,1)} = \int_0^\infty (1 - G_{u1}(t))(1 - F_{d2}(t))dt, \\ h_{(1,0)} = \int_0^\infty (1 - G_{d1}(t))(1 - F_{u2}(t))dt, \\ h_{(0,0)} = \int_0^\infty (1 - G_{u2}(t))(1 - G_{d2}(t))dt, \\ h_{(0,1')} = h_{(0,0')} = 0. \end{cases} \tag{7}$$

The steady-state probability π_i for the SMP is given by

$$\pi_i = \frac{v_i h_i}{\sum_j v_j h_j}. \tag{8}$$

D. Conservative Retraining Model

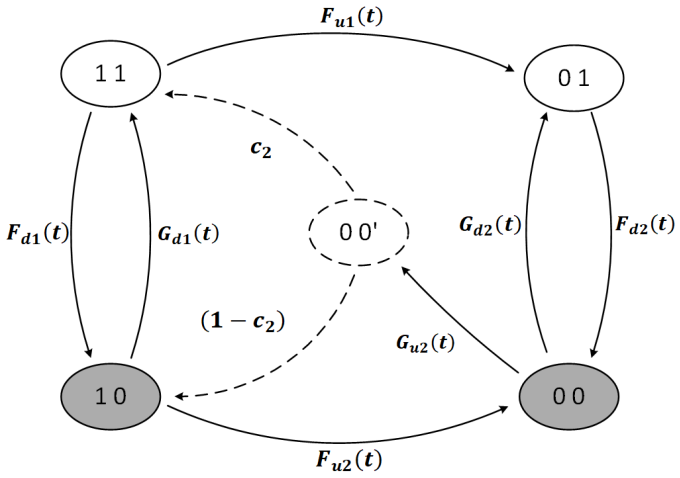


Fig. 4: A SMP depicting the dynamics of MLS under a conservative retraining policy. The sojourn time of the transitions caused by dataset shift is set to follow general functions $F_{u1}(t)$, $F_{u2}(t)$, $F_{d1}(t)$, and $F_{d2}(t)$. The sojourn time of the transitions caused by retraining is also set to follow some general distribution function. $G_{u2}(t)$ denotes the sojourn time distribution from $(0,0)$ to $(0,0')$. $G_{d1}(t)$ denotes the sojourn time distribution from $(1,0)$ to $(1,1)$. $G_{d2}(t)$ denotes the sojourn time distribution from $(0,0)$ to $(0,1)$. $(0,0')$ is a vanishing states with sojourn time of 0. c_2 denotes the possibility that once $(0,0')$ is entered, it will be transited to $(1,1)$.

The two components MLS under the conservative retraining policy is modeled by the SMP as shown in Fig. 4. At states (1,1) and (0,1), no retraining is attempted because the performance of the downstream model satisfies the threshold. At State (1,0), the downstream model would be retrained. At State (0,0), both upstream and downstream models can be retrained. Compared to the system in Fig. 4, because there would be no retraining happening in state (0,1), the state (0,1') is also absent. The transition from (0,1') to (1,1) is removed, which is a potential loss. However, at the same time, the transition from (0,1') to (1,0), which is the Type 1 entangle enhancement that causes the system to fail, is also removed, which might benefit the system.

Because the embedded DTMC of the SMP satisfies the homogeneity property, we can obtain the steady-state probability vector $\boldsymbol{v}$ by $\boldsymbol{P}$ using the formula $\boldsymbol{P} = \lim_{t\to\infty} \boldsymbol{K}(t)$ again,

$$\boldsymbol{P} = \begin{array}{c} \begin{matrix} (1,1) & (0,1) & (1,0) & (0,0) & (0,0') \end{matrix} \\ \begin{pmatrix} 0 & P_1 & 1-P_1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ P_2 & 0 & 0 & 1-P_2 & 0 \\ 0 & P_3 & 0 & 0 & 1-P_3 \\ c_2 & 0 & 1-c_2 & 0 & 0 \end{pmatrix} \end{array}, \tag{9}$$

$$\begin{cases} P_1 = \int_0^\infty (1 - F_{d1}(t))dF_{u1}(t), \\ P_2 = \int_0^\infty (1 - F_{u2}(t))dG_{d1}(t), \\ P_3 = \int_0^\infty (1 - G_{u2}(t))dG_{d2}(t). \end{cases} \tag{10}$$

After computing the steady-state probability $\boldsymbol{v}$, we derive the mean sojourn time of state i h_i:

$$\begin{cases} h_{(1,1)} = \int_0^\infty (1 - F_{u1}(t))(1 - F_{d1}(t))dt, \\ h_{(0,1)} = \int_0^\infty (1 - F_{d2}(t))dt, \\ h_{(1,0)} = \int_0^\infty (1 - G_{d1}(t))(1 - F_{u2}(t))dt, \\ h_{(0,0)} = \int_0^\infty (1 - G_{u2}(t))(1 - G_{d2}(t))dt, \\ h_{(0,0')} = 0. \end{cases} \tag{11}$$

Steady-state probability π_i for the SMP state i can be derived from Equation 8.

E. Metrics Definition

To compare the effectiveness of the system under different policies, we define *service availability* and *service continuity* as evaluation metrics and provide the formulas for calculating these metrics.

Specifically, under the policy k, we define U_k as the set that comprise all the states that are deemed as available states, D_k as the set that comprise all the states that are deemed as unavailable states and π_i as the steady-state availability of state i, which can be derived by Equation 8.

The service availability under the policy k is defined as:

$$A_k = \prod_{i \in U_k} \pi_i. \tag{12}$$

The metric reflects the long-term probability that the system meets user satisfaction by reaching the threshold when users utilize the MLS.

The service continuity is defined as the average duration between transitions that happen from an available state ((1,1), (0,1), or (0,1')) to an unavailable state ((1,0), (0,0), or (0,0')). The higher the service continuity, the more stable the system under that policy is, as it implies a longer duration between failure events. Formally, service continuity under policy k can be defined as the inverse of the sum of throughput from the available state set U_k to the unavailable state set D_k. We define $tp_{i,j}$ as the throughput from state i to state j. To calculate $tp_{i,j}$, we first derive the expected duration between system entering state i and system entering state i again, denoted by W_i:

$$W_i = \frac{\sum_k v_k h_k}{v_i}. \tag{13}$$

The derivation of the expected return duration was proved in Theorem 6.8.1 in [49]. Then the frequency of state i in the long run is exactly $\frac{1}{W_i}$. By multiplying the transition probability of the embedded DTMC from state i to state j, we derive the $tp_{i,j}$:

$$tp_{i,j} = \frac{1}{W_i} * p_{i,j}. \tag{14}$$

And the service continuity C_k can be mathematically expressed as:

$$C_k = \frac{1}{\sum_{i \in U_k, j \in D_k} tp_{i,j}}. \tag{15}$$

VI. Numerical Analysis

A. Research Questions

In this section, we perform numerical analyses on the proposed SMPs to investigate the effectiveness of retraining policies in terms of service availability and service continuity. Specifically, the analysis aims to address the following research questions:

- RQ1: What factors influence service availability and service continuity under different policies?
- RQ2: How does the shape of the distribution function of the state transition time impact the analysis results?
- RQ3: How is service availability correlated or in trade-off relation with service continuity under different policies?

We also present a strategy for selecting the appropriate retraining policies to meet the requirements of the MLS.

B. Parameter Assignment using Weibull Holding Time

TABLE V: Sojourn Time Distribution

Sojourn time distribution	Function or value	Transition states
$F_{u1}(t)$	$1 - exp[-(\eta_1 t)^1]$	$(1,1) \to (0,1)$
$F_{u2}(t)$	$1 - exp[-(\eta_1 t)^1]$	$(1,0) \to (0,0)$
$F_{d1}(t)$	$1 - exp[-(\eta_2 t)^1]$	$(1,1) \to (1,0)$
$F_{d2}(t)$	$1 - exp[-(\eta_2 t)^1]$	$(0,1) \to (0,0)$
$G_{u1}(t)$	$1 - exp[-(\eta_3 t)^{\beta_1}]$	$(0,1) \to (0,1')$
$G_{u2}(t)$	$1 - exp[-(\eta_3 t)^{\beta_1}]$	$(0,0) \to (0,0')$
$G_{d1}(t)$	$1 - exp[-(\eta_4 t)^{\beta_2}]$	$(1,0) \to (1,1)$
$G_{d2}(t)$	$1 - exp[-(\eta_5 t)^{\beta_3}]$	$(0,0) \to (0,1)$

TABLE VI: Parameter Input

Parameter	Value	Description
η_1	0.0125	Scale parameter of $F_{u1}(t)$, $F_{u2}(t)$
η_2	0.025	Scale parameter of $F_{d1}(t)$, $F_{d2}(t)$
η_3	0.057	Scale parameter of $G_{u1}(t)$, $G_{u2}(t)$
η_4	0.112	Scale parameter of $G_{d1}(t)$
η_5	0.011	Scale parameter of $G_{d2}(t)$
β_1	2	Shape parameter of $G_{u1}(t)$, $G_{u2}(t)$
β_2	3	Shape parameter of $G_{d1}(t)$
β_3	3	Shape parameter of $G_{d2}(t)$
c_1	0.75	Coverage factor for type 1 entangled enhancement
c_2	0.5	Coverage factor for type 2 entangled enhancement

Table V and Table VI present the input parameter functions and values used in the analysis. Since SMP allows general distributions for all the state transition times, we employ the Weibull distribution as it can describe the transition behavior with increasing, constant, and decreasing hazard rates. We use 2-parameter Weibull distributions to model the sojourn time distribution, which consists of 2 parameters: the scale parameter η and the shape parameter β. The scale parameter η locates the position of the probability density function (PDF) of the distribution on the time axis. As η increases, the entire distribution shifts to the left, which means that the time to state transition becomes shorter. For a larger η, recoveries or failures are expected to happen more quickly. The shape parameter β determines the shape of the PDF and is closely related to the hazard rate of the distribution. When $\beta < 1$, the hazard rate is monotonically decreasing, and when $\beta > 1$, the hazard rate is monotonically increasing. As β increases from 1 to higher values, the peak of the PDF becomes more pronounced and shifts to the right. This would generally increase the expected time to state transition, but at the same time makes such transition more deterministic and predictable, as the recovery/failure time clusters around a certain period.

The parameter values in Table VI are chosen from the parameter space satisfying the following conditions:

$$\eta_1 < \eta_2, \tag{16}$$

$$\int_0^\infty \exp[-(\eta_4 t)^{\beta_2}]\, d(t) < \int_0^\infty \exp[-(\eta_3 t)^{\beta_1}]\, d(t), \tag{17}$$

$$\int_0^\infty \exp[-(\eta_3 t)^{\beta_1}]\, d(t) < \int_0^\infty \exp[-(\eta_5 t)^{\beta_3}]\, d(t), \tag{18}$$

$$c_1 > c_2. \tag{19}$$

For conditions 16 and 17, given the cascading effect of upstream model recoveries on all downstream models, the upstream model is presumed to exhibit greater robustness against dataset shift and consequently undergoes recoveries less frequently. Therefore, the mean time to failure (MTTF) and the mean time to recovery (MTTR) of the upstream model should be larger than those of the downstream model.

For condition 18, the direct update of the downstream model resulting in a restoration in system performance while the upstream model still maintains a deteriorated performance is

TABLE VII: Scaled Sensitivity for Availability and Continuity

Parameter θ	$SS_\theta(A_p)$	$SS_\theta(A_c)$	$SS_\theta(C_p)$	$SS_\theta(C_c)$
η_1	-0.0643	-0.0601	0.00383	0.0601
η_2	-0.201	-0.205	-0.700	-0.787
η_3	0.0811	0.166	-0.112	-0.166
η_4	0.181	0.0956	-0.181	-0.0955
η_5	-0.000900	0.00205	-0.00182	-0.00205
β_1	-0.000302	0.00168	0.00303	-0.00168
β_2	-0.0102	-0.00625	0.0102	0.00625
β_3	-0.00125	-0.00286	0.00241	0.00286
c_1	0.0413	-	0.169	-
c_2	0.0212	0.0253	-0.0212	-0.0253

very unlikely. Therefore, the MTTR of this process is the largest compared to other recovery processes.

For condition 19, c_2 represents the possibility of a more demanding condition, requiring the upstream model's recovery, which improves the downstream model's performance metric from below τ_d to above τ_d. In comparison, c_1 only represents the possibility that the upstream model's recovery does not negatively affect the downstream model to the point where it fails to reach the threshold, which is a condition that is more likely to occur. Therefore, c_1 is likely to be larger than c_2.

C. RQ1: Sensitivity Analysis

The first two columns in Tables VII show the scaled sensitivities of different parameters influencing A_p and A_c, which are the service availability of the system under the progressive retraining policy and the conservative retraining policy, respectively. For both A_p and A_c, the three most important parameters are η_2, η_4 and η_3, which represent the scale parameters of the Weibull sojourn time of the transactions due to downstream model dataset shift, downstream model update and upstream model update. It is obvious that the scale parameter related to the distribution of the dataset is negatively related to availability, while the scale parameter related to the distribution of recovery is positively related to availability.

The last two columns in Table VII show the sensitivities of various parameters influencing C_p and C_c, which are the service continuity of the system under progressive retraining policy and conservative retraining policy, respectively. The three most important parameters affecting service continuity are again η_2, η_3, and η_4. An interesting phenomenon is that most parameters have opposite effects on availability and continuity. For example, increasing η_3 and η_4 would increase the availability at the cost of continuity, regardless of the policy choice. This finding implies that increasing the frequency of the model retraining could improve availability but reduce the continuity.

Findings: Reducing downstream model degradation is the most effective method to increase both availability and continuity. Besides, increasing the frequency of model retraining to gain more service availability has the potential risk of harming service continuity.

D. RQ2: Comparison to Baseline

In previous studies, only one used CTMC for availability analysis [1]. The same analysis can be replicated using our model by setting the shape parameter of the Weibull distribution of the recovery process to 1, making the transition time exponentially distributed. However, in reality, recovery times may follow various distributions. To demonstrate the impact of different distribution assumptions, we vary the shape parameter of the Weibull distribution while keeping the MTTR constant. Figure 5 illustrates how the shape parameter affects service availability.

The figures show that even with a fixed MTTR, increasing the shape parameter β for the recovery process results in a decrease in availability. This can be explained by the behavior of the PDF of the distribution: when β is small, the PDF is more concentrated on the left, indicating a higher likelihood of quick recovery. As β increases, the PDF spreads out and shifts to the right, leading to a more consistent recovery time, thus possibly reducing service availability.

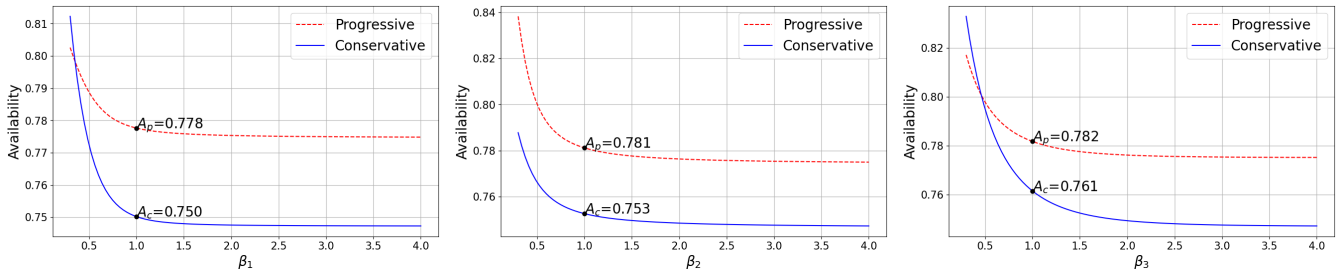
Furthermore, an interesting trend is observed in Figure 5 (a) and (c). Initially, the progressive retraining policy shows lower availability compared to the conservative retraining policy. However, as the shape parameter increases, the progressive policy eventually surpasses the conservative policy in terms of availability.

These observations cannot be captured by CTMC modeling, where the hazard rate remains constant (with a shape parameter equal to 1), as depicted by the two points on each figure. Therefore, the previous model had a shortcoming in accurately estimating availability due to the exponential assumption. In contrast, the proposed SMP can accommodate diverse recovery time distributions with different shapes, allowing for a more comprehensive analysis of MLS.

Findings: Despite the same MTTR, the shape of the PDF for the recovery process would influence the system availability and even the preference in policy choice. In this aspect, SMP provides a more comprehensive analysis than CTMC.

E. RQ3: Trade-off analysis: Availability vs Continuity

In order to analyze the trade-off between availability and continuity, we further plot service availability against service continuity under different retraining policies by varying individual parameters, as shown in Fig. 6. In Fig. 6 (a), increasing the scale parameter for downstream model failure reduces both service availability and service continuity, regardless of the policy. Figures 6 (b) and (c) illustrate the relationship between service availability and the scale parameters η_3 for upstream model recovery and η_4 for downstream model recovery (when only the downstream model fails while the upstream model works). An increase in these parameters enhances service availability but decreases service continuity. This aligns with the results in Table VII.

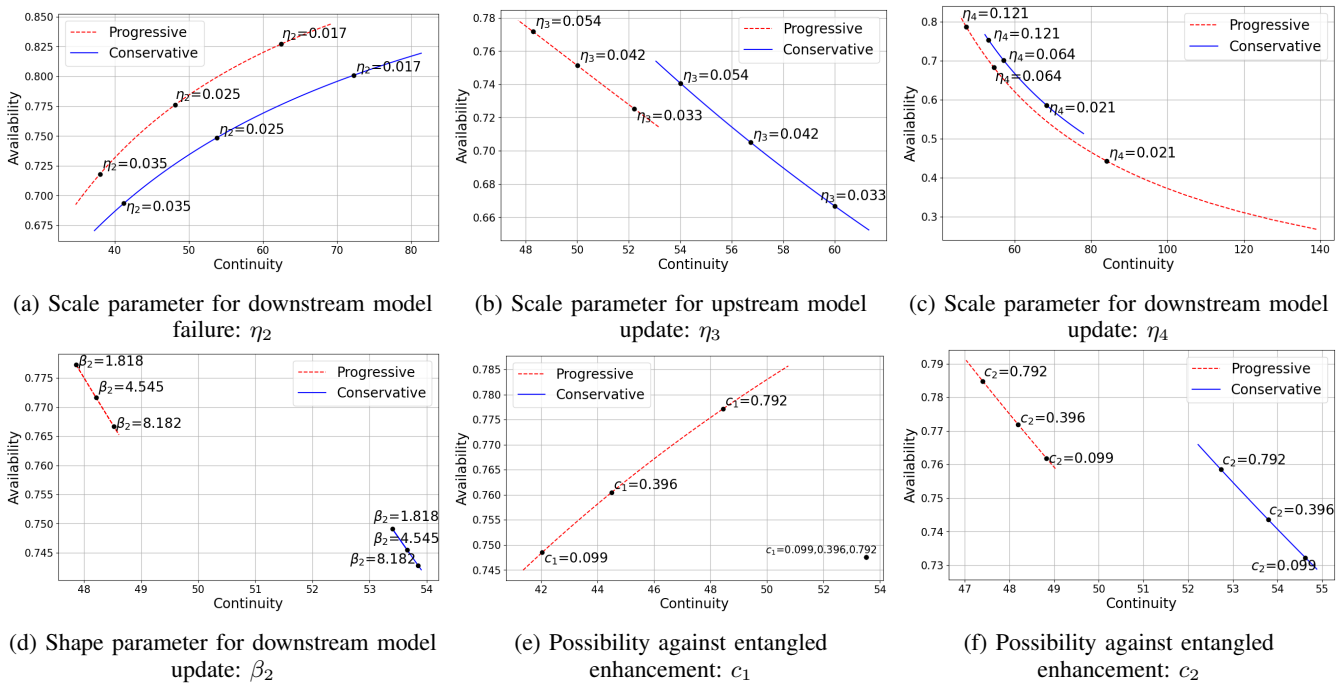


(a) Shape Parameter for upstream model update: β_1

(b) Shape Parameter for downstream model update: β_2

(c) Shape Parameter for downstream model update: β_3

Fig. 5: Service Availability regarding to different shape parameters



(a) Scale parameter for downstream model failure: η_2

(b) Scale parameter for upstream model update: η_3

(c) Scale parameter for downstream model update: η_4

(d) Shape parameter for downstream model update: β_2

(e) Possibility against entangled enhancement: c_1

(f) Possibility against entangled enhancement: c_2

Fig. 6: Service Availability and service continuity regarding different parameter changes

Further examination of the figures reveals a mostly consistent trend wherein the system operating under the progressive retraining policy exhibits higher availability compared to the conservative retraining policy. At the same time, the system operating under a conservative retraining policy consistently exhibits a higher continuity in comparison to the system employing a progressive retraining policy, given the same parameters. The result implies that, under the progressive retraining policy, the system, on average, is more available than the system under the conservative retraining policy. On the other hand, under the conservative retraining policy, transitions from an available state to an unavailable state occur less frequently. Consequently, there is a trade-off between service availability and service continuity in choosing different policies. When prioritizing service continuity, the conservative retraining policy is deemed more favorable. For service availability, the progressive retraining policy is preferable.

Findings: The maintenance policies can confront the significant trade-off between service availability and service continuity. Therefore, choosing one policy typically enhances one aspect at the expense of the other.

VII. Strategy Analysis

The trade-off analysis results can guide the maintenance of the MLS, which includes a strategy to mitigate imperfect retraining. We found that in most cases, the MLS faces the trade-off between service availability and service continuity. Given a specific use case of the MLS, either service availability or service continuity may be more vital than the other metric. For example, ML tasks that prioritize high long-term average accuracy, while tolerating occasional performance drops, should focus on service availability rather than service continuity. Predictive maintenance is one key example of such a task. It is a continuous process that requires regular

updates and refinements to adapt to changing equipment and evolving operational conditions [41]. Industry practices in predictive maintenance emphasize improving the interpretability of anomaly detection for unsupervised methods and generating more labeled data for supervised methods to enhance long-term accuracy. This approach prioritizes building robust models that perform well over extended periods, rather than maintaining stability during specific short-term intervals [40]. Therefore, ML engineers working on predictive maintenance need a higher service availability. Conversely, some ML tasks demand high run-time accuracy, where model updates that could compromise performance should be avoided. In these scenarios, service continuity is more critical. One example is ML systems for patient care. While such systems can benefit from periodic updates using data collected from patients to improve diagnostic accuracy or support better management decisions [45], frequent retraining poses risks [43]. In such cases, service continuity tends to be prioritized to ensure consistent and reliable real-time performance, safeguarding patient trust and safety.

To ensure the MLS effectively achieves its goal, it is crucial to select the right retraining policy. From the trade-off analysis in Figure 6 (e), we observe that when the possibility of entangled enhancement becomes very large, which is represented by the value of c_1 approaches to 0 (for example, $c_1 = 0.1$), the progressive retraining policy should not be applied as it is inferior to conservative retraining policy in both service availability and service continuity. On the other hand, from Figure 6 (c), it can be observed that if the successful retraining of the downstream model happens rarely (for example, $\eta_4 = 0.06$), the conservative retraining policy should be applied. Based on these observations, the system engineers may derive a simple strategy as presented in Figure 7. Although the system parameter values differ in other

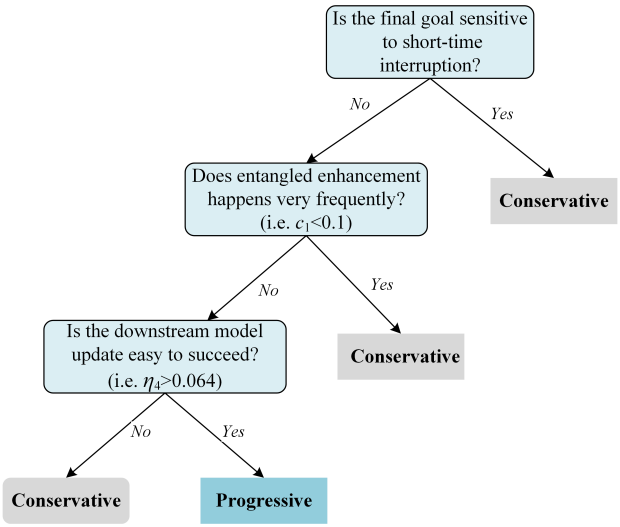


Fig. 7: A strategy to select the retraining policy for the MLS

application scenarios, the strategy for choosing the retraining policy can also be applied in these scenarios.

VIII. Limitations

In this study, we focused on a simple MLS comprising two dependent components, where the system's output is determined primarily by the downstream model. While this configuration can effectively capture the essential dynamics of imperfect retraining caused by component entanglement, the real-world MLSs may involve more than two components. Such configurations could introduce additional layers of complexity, including interdependencies and cascading effects, which are not addressed in this work.

This study evaluates the system using service availability and service continuity. While these metrics provide valuable insights, additional criteria could be considered in future work. For instance, retraining operations incur computational costs, and in a large-scale MLS, multiple modules may require retraining simultaneously. Ensuring that the retraining units have sufficient capacity to handle these demands without disrupting overall system operations could be a critical area for further investigation.

IX. Conclusion

This paper sheds light on an unattended issue of imperfect retraining in MLS operations. Our experiments in 3D object detection and sentence classification systems demonstrated the real problems of entangled enhancement. We use SMP as our state-space modeling method, allowing a general distribution to the transition process. We introduced service continuity as a metric to evaluate the stability of MLS. We demonstrate that the maintenance policies under entangled enhancement face a notable trade-off between service availability and continuity, as revealed through numerical analysis, and introduce a strategy guide to aid in choosing the optimal maintenance policy.

Future research can explore additional retraining policies, refine modeling techniques, and apply our findings to a broader range of application domains. We plan to extend the analysis to MLS with more components and develop a general model for MLS containing N components.

References

[1] Z. Wang and F. Machida, "Maintaining performance of a machine learning system under imperfect retraining" in 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC), 2024.

[2] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," Science, vol. 349, no. 6245, pp. 255–260, 2015.

[3] R. Wu, C. Guo, A. Hannun, and L. van der Maaten, "Fixes that fail: Self-defeating improvements in machine-learning systems," in Advances in Neural Information Processing Systems, vol. 34, pp. 11745–11756, 2021.

[4] F. Yu, D. Wang, L. Shangguan, M. Zhang, X. Tang, C. Liu, and X. Chen, "A survey of large-scale deep learning serving system optimization: Challenges and opportunities," In arXiv:2111.14247, 2022.

[5] H. Fang, S. Gupta, F. Iandola, R. K. Srivastava, L. Deng, P. Dollãr, J. Gao, X. He, M. Mitchell, J. C. Platt et al., "From captions to visual concepts and back," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1473–1482, 2015.

[6] L. Bottou, J. Peters, J. Quiñonero-Candela, D. X. Charles, D. M. Chickering, E. Portugaly, D. Ray, P. Simard, and E. Snelson, "Counterfactual reasoning and learning systems: The example of computational advertising." Journal of Machine Learning Research, vol. 14, no. 11, pp. 3207-3260, 2013.

[7] J. Hron, K. Krauth, M. Jordan, and N. Kilbertus, "On component interactions in two-stage recommender systems," Advances in neural information processing systems, vol. 34, pp. 2744–2757, 2021.

[8] J. G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. V. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," Pattern Recognition, vol. 45, no. 1, pp. 521-530, 2012.

[9] Y. Lyu, H. Li, M. Sayagh, Z. M. J. Jiang, and A. E. Hassan, "An empirical study of the impact of data splitting decisions on the performance of aiops solutions," ACM Transactions on Software Engineering and Methodology, vol. 30, no. 4, 2021.

[10] J. Quiñonero Candela, M. Sugiyama, A. Schwaighofer and N.D. Lawrence, Dataset Shift in Machine Learning. The MIT Press, pp. 3–28, 2009.

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," International Journal of Computer Vision, vol. 88, no. 2, pp. 303–338, 2010.

[12] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p. 226–235, 2003.

[13] H. Wang, P. S. Yu, and J. Han, "Mining Concept-Drifting Data Streams, Data Mining and Knowledge Discovery Handbook", pp. 789–802, 2010.

[14] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Na- gappan, B. Nushi, and T. Zimmermann, "Software engineering for machine learning: A case study," in 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), pp. 291–300, 2019.

[15] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5410–5418, 2018.

[16] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–779, 2019.

[17] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in 2012 IEEE conference on computer vision and pattern recognition. IEEE, pp. 3354–3361, 2012.

[18] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data management challenges in production machine learning," in Proceedings of the 2017 ACM International Conference on Management of Data, pp. 1723–1726, 2017.

[19] R. Kocielnik, S. Amershi, and P. N. Bennett, "Will you accept an imperfect AI? exploring designs for adjusting end-user expectations of AI systems," in Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, pp. 1-14, 2019.

[20] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8445–8453, 2019.

[21] N. Hewage and D. Meedeniya, "Machine Learning Operations: A Survey on MLOps Tool Support", In arXiv:2202.10169, 2022.

[22] A. Simonelli, S. R. Bulo, L. Porzi, M. López-Antequera, and P. Kontschieder, "Disentangling monocular 3d object detection," in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1991–1999, 2019.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186. 2019.

[24] Y. Kim, "Convolutional neural networks for sentence classification," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1746–1751, 2014.

[25] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp. 785–794, 2016.

[26] W. Li, S. Gao, H. Zhou, Z. Huang, K. Zhang, and W. Li, "The automatic text classification method based on bert and feature union," in 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), pp. 774–777, 2019.

[27] A. C. Mazari, N. Boudoukhani, and A. Djeffal, "Bert-based ensemble learning for multi-aspect hate speech detection," Cluster Computing, vol. 27, no. 1, pp. 325–339, 2024.

[28] S.-Y. Lin, Y.-C. Kung, and F.-Y. Leu, "Predictive intelligence in harmful news identification by bert-based ensemble learning model with text sentiment analysis," Information Processing & Management, vol. 59, no. 2, p. 102872, 2022.

[29] B. Karlaš, M. Interlandi, C. Renggli, W. Wu, C. Zhang, D. Mukunthu Iyappan Babu, J. Edwards, C. Lauren, A. Xu, and M. Weimer, "Building continuous integration services for machine learning," in Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2407–2415, 2020.

[30] D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine learning operations (mlops): Overview, definition, and architecture," IEEE Access, vol. 11, pp. 31866–31879, 2023.

[31] B. Nushi, E. Kamar, E. Horvitz, and D. Kossmann, "On human intellect and machine failures: Troubleshooting integrative machine learning systems," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31, no. 1, pp. 1017-1025, 2017.

[32] A. D'Amour, K. Heller, D. Moldovan, B. Adlam, B. Alipanahi, A. Beutel, C. Chen, J. Deaton, J. Eisenstein, M. D. Hoffman et al., "Underspecification presents challenges for credibility in modern machine learning," The Journal of Machine Learning Research, vol. 23, no. 1, pp. 10237–10297, 2022.

[33] K. S. Trivedi and A. Bobbio, Reliability and availability engineering: modeling, analysis, and applications. Cambridge University Press, 2017.

[34] M. Tortorella, "Service reliability theory and engineering I: Foundations," Quality Technology & Quantitative Management, vol. 2, no. 1, pp. 1–16, 2005

[35] A. Goyal, S. Lavenberg, and K. Trivedi, "Probabilistic modeling of computer system availability," Annals of Operations Research, vol. 8, pp. 285–306, 1987

[36] Y. Liu and K. Trivedi, "Survivability quantification: The analytical modeling approach," International Journal of Performability Engineering, vol. 2, no. 1, p. 29, 2006.

[37] L. Yin, R. Fricks, and K. Trivedi, "Application of semi-Markov process and CTMC to evaluation of UPS system availability," in Proc. Reliability and Maintainability Symp., pp. 584–591, 2002.

[38] K. Vaidyanathan and K. Trivedi, "A comprehensive model for software rejuvenation," IEEE Transactions on Dependable and Secure Computing, vol. 2, no. 2, pp. 124–137, Apr. 2005.

[39] V. J. Girish Kumar and O. P. Gandhi, "Availability analysis of repairable mechanical systems using analytical semi-markov approach," Quality Engineering, vol. 25, no. 2, pp. 97–107, 2013.

[40] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," Advances in neural information processing systems, vol. 28, pp. 2503-2511, 2015.

[41] J. Hurtado, D. Salvati, R. Semola, M. Bosio, and V. Lomonaco, "Continual learning for predictive maintenance: Overview and challenges," Intelligent Systems with Applications, vol. 19, p. 200251, 2023.

[42] P. Nunes, J. Santos, and E. Rocha, "Challenges in predictive maintenance – a review," CIRP Journal of Manufacturing Science and Technology, vol. 40, pp. 53–67, 2023.

[43] C. S. Lee and A. Y. Lee, "Clinical applications of continual learning machine learning," The Lancet Digital Health, vol. 2, no. 6, pp. e279–e281, 2020.

[44] C. F. G. D. Santos and J. P. Papa, "Avoiding overfitting: A survey on regularization methods for convolutional neural networks," ACM Computing Surveys (CSUR), vol. 54, no. 10, pp. 1–25, 2022.

[45] K. N. Vokinger, S. Feuerriegel, and A. S. Kesselheim, "Continual learning in medical devices: Fda's action plan and beyond," The Lancet Digital Health, vol. 3, no. 6, pp. e337–e338, 2021.

[46] J. Antony, D. Jalušić, S. Bergweiler, Á. Hajnal, Žlabravec, M. Emődi, D. Strbad, T. Legler, and A. C. Marosi, "Adapting to changes: A novel framework for continual machine learning in industrial applications," Journal of Grid Computing, vol. 22, no. 4, pp. 1–19, 2024.

[47] T. Vakili, A. Lamproudis, A. Henriksson, and H. Dalianis, "Downstream task performance of BERT models pre-trained using automatically de-identified clinical data," in Proceedings of the 13th Language Resources and Evaluation Conference. pp. 4245–4252, 2022.

[48] H. Choi, J. Kim, S. Joe, and Y. Gwon, "Evaluation of bert and albert sentence embedding performance on downstream nlp tasks," in 2020 25th International Conference on Pattern Recognition (ICPR), pp. 5482–5487, 2021.

[49] R. G. Gallager, Discrete Stochastic Processes, Springer, Boston, MA, 1996.