Maintaining Performance of a Machine Learning System Against Imperfect Retraining

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 (MLS) often consist of diverse machine learning models to attain demanding and complex objectives. Despite their advanced functionalities, MLSs encounter the inevitable challenge of performance deterioration caused by distribution changes often referred to as dataset shift. When models encountering dataset shift, retraining machine learning model with new datasets is essential to restore the performance. However, model retraining is not always perfect due to component entanglement, resulting in failures to maintain the required level of performance. To address this issue, this paper investigates the impact of imperfect retraining on the overall performance of MLSs, and accordingly propose two maintenance policies, progressive and conservative retraining policies. We consider an MLS consisting of two sequentially-dependent machine learning models and develop continuous-time Markov chains capturing the dynamics of performance degradation and retraining of machine learning models. The results of parametric sensitivity analysis demonstrate that the progressive retraining policy and conservative retraining policy could provide higher service availability in different conditions.**

***Index Terms*—Markov chain, dataset shift, machine learning system, retraining, service availability**

I. Introduction

Machine learning (ML) is a statistical approach to learning patterns from data and generalize to unseen data, thus allowing the need for machine to perform tasks based on the acquired patterns without explicit instructions. In recent years, ML has been an important technique to help make prediction, classification and grouping, and thus plays an important role in computer vision, natural language processing and autonomous driving. Machine learning system (MLS) is a system that incorporates ML models as its components. MLS allows large teams to effectively divide-and-conquer complex engineering tasks by making each team focusing on a specific component. Examples of MLS includes an image captioning system, a search advertising system, and a personalized recommendation system [12] [13] [16].

Within the operation of MLSs, each component of ML model might experience a deterioration in performance due to dataset shift [1], a phenomenon in which the change in data distribution worsen the degraded ML models' ability to perform the task. When encountering dataset shift in the operation, retraining the ML models is essential to maintain the performance. However, we witness that retraining can be imperfect due to several reasons. For instance, ML components might experience component entanglement [3], biased selection of training and validation data, or over-fitting due to inappropriate training strategy [15].

In this study, we aim to investigate the impact of imperfect retraining related to component entanglement. When prediction performance is deteriorated due to dataset shift, individual ML models in MLS cannot be retrained jointly in many cases. For example, some models may have been developed by cloud providers and thus cannot be changed, or it is too time-consuming to jointly retrain the whole system [3] [9]. This could lead to the problem of entangled enhancement, indicating that the improvement of a certain component by retraining does not result in the improvement in the entire performance of the MLS. To counteract the issue of entangle enhancement, MLS requires a careful maintenance operation to achieve higher service availability considering the risk of imperfect retraining. However, there are very few studies making effort to describe MLSs from a system maintenance perspective and analyzing the service availability.

To narrow the gap, we propose and examine two maintenance policies for retraining ML models in an MLS, namely *progressive retraining* and *conservative retraining* policies. The progressive retraining policy retrains ML models whenever the performance of a component in MLS fails to achieve the required standard, while the conservative retraining policy only retrains the ML models when the overall performance of the system is deteriorated. To compare the effectiveness of the retraining policies, we develop Continuous-time Markov Chains (CMTCs) that can capture the state transitions of an MLS consists of two sequentially-dependent ML models. Through the comprehensive sensitivity analysis on the proposed models, we find that the conservative retraining policy and the progressive retraining policy have advantages in terms of system availability in different conditions.

To sum up, the paper makes the following contributions:

- We formulate the problem of imperfect retraining encountered in operations of MLSs.
- We propose two maintenance policies, the progressive and the conservative retraining policies, and evaluate the effectiveness through the analysis of CTMCs representing the state transitions of the MLS.

The rest of this paper is organized as follows: Section II presents the related work. Section III explains the setting of the problem in detail. Section IV proposes the availability modelling of the MLS using CTMC. Section V conducts numerical analysis on the availability models of the system. Finally, Section VI concludes the paper.

II. Related Work

A. Dataset Shift

Dataset shift, also known as concept drift, is an issue that has been widely studied in the ML community. It was first defined as "cases where the joint distribution of inputs and outputs differs between training and test stage" [4]. Later, many other definitions were proposed [5] [6] [7]. In this paper, we adopt a more unified and comprehensive definition, which is, for any situation in which training and test data follow distributions that are in some way different [1]. Dataset shift can be due to either sample selection bias, which is a bias caused by training set being selected non-uniformly from the population, or non-stationary environment, in which real-world data is not stationary in the sense of time and space [1]. Regardless of the reason, dataset shift becomes a main factor threatening the performance of the MLS over the time, and any remedies to recover the performance is necessary for long-run operation.

B. Machine Learning Component Entanglement

There are different aspects with regard to ML component entanglement that have been studied. In a component-based MLS, when the quality of a component depends on the output of previous components, in time of system failure, sometimes blame cannot be assigned to individual components because it is not possible to disentangle their individual impact on the final error [3]. A self-defeating improvement is a unique problem observed in a component-based MLS. Under the assumption of the independent training, an improvement to the upstream model might result in no change or even deterioration in at least one of its downstream models [2]. This is dramatically different from software system, in which improvement to a certain component usually would guarantee to improve the whole system [10].

C. Countering Imperfect Retraining

Existing studies on MLS against imperfect retraining either focuses on proposing a more reliable ML algorithm [1], providing a monitoring system by introducing the human factor to reduce the potential failure in training [3], or trying to diagnose the root of the failure and made corresponding adjustment to the model themselves [2]. In this study, instead of focusing on the design of algorithm or post-processing to prevent imperfect retraining, we focus on minimizing such failure's influence for the MLS. This approach shared resemblance to software system failure study such as [11], but handling failure in training for MLS is distinct from handling software system failure. Unlike software system, each component in the MLS lacks explicit functionality [10], therefore an update to a certain component cannot guarantee the improvement of the whole system.

III. Problem Setting

A. Machine Learning System

We consider an MLS consisting of multiple ML models as a static directed acyclic graph (DAG), $G = (\mathcal{V}, \mathcal{E})$, where a vertex $v \in \mathcal{V}$ represents an ML model and an edge $(v, w) \in \mathcal{E}$ represents the output of model v being used by model w. Given an edge (v, w), v is called the *upstream model* and w is called the *downstream model*. Fig. 1 shows an example MLS with an upstream model u and a downstream model d. Each vertex in G has a function that applied on the output and data from its upstream model. We denote this function for vertex v as $f_v(x)$. For the parent node u which has no upstream model, it only takes the data relevant for training from the input original data X, we denote this part of the data as $X^{(u)}$. As it can be seen in the figure, a subset of the original input X, X_u, is used for the upstream model to process output $f_u(X_u)$. the subset of the original input X, X_d, and the output from upstream model $f_u(X_u)$, would be the input for downstream model. The system's output can be represented by $f_d(f_u(X_u + f_d(X_d))$.

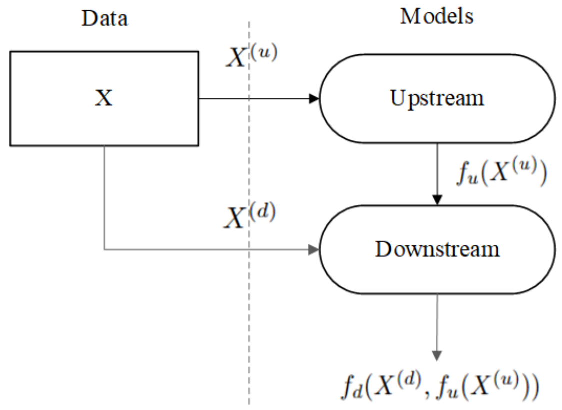


Fig. 1: An MLS with two components $\mathcal{V} = \{u, d\}$, where u is upstream of d, and the output for the system becomes $f_d(X^{(d)}, f_u(X^{(u)}))$.

An example of such system was presented in [2]. The proposed system consists of an upstream depth estimation model to infer the z-coordinate from 2D images, and a downstream object detection model that uses 3D information to detect the location of cars in 3D coordinates. After the retraining and update of the upstream model, which results in an overall decreased mean error in distance of z-coordinate, the downstream car detection model experiences performance deterioration in the detection accuracy in 3D space.

a) Performance: In an MLS, we assume that the performance of the model is periodically evaluated by an ideal hypothesis dataset, denoted as X_{real}, that faithfully represents the latest real-world dataset, and contains data points $\{(x'_1, y'_1), ...(x'_{n_2}, y'_{n_2})\}$. The performance metric could be Precision, Recall or Mean Average Error, depending on the type of the task. Each ML model is required to satisfy a certain level of of the performance that we refer to the threshold to the target metric. Although the performance can be deteriorated over the time due to dataset shift, we assume that the threshold can be met eventually throughout multiple trials of retraining.

b) Dataset Shift: Initially, both upstream model u and downstream model d satisfy the requirements, therefore showing capability of handling real-world task. As the time goes by, the hypothetical dataset that represents the real-world distribution, X_{real}, changes because of the dataset shift. This process causes a gradually worsening effect for both upstream and downstream models on X_{real}, until either of those to drop below the threshold, thus be denoted as a failure to that specific component.

c) Training: For the training process, a dataset X_{train} that consists of data points $\{(x_1, y_1), ..., (x_{n_1}, y_{n_1})\}$ is used. For each model v, algorithm f_v is chosen from the hypothesis set H_v based on the X_{train}. f_v is trained on the dataset X_{train}. The learning algorithm f_v, set X_{train} and hypothesis set H_v is assumed to be unchanged during the training process. Training is conducted separately, and hence means that during the training process of d, the result is not backward propagated to upstream model u.

B. Imperfect retraining and self-defeating improvement

a) Imperfect retraining: The primary focus of our investigation centers on understanding the impact of imperfect retraining on the overall performance of an MLS. To counter the effect of dataset shift, usually retraining is necessary to maintain the performance of the MLS. However, retraining for a specific component might not be perfect and can even worsen the performance of the system. We argue that, there are mainly two types of imperfect retraining:

1) Degenerative retraining: The retraining of the model makes that component perform even worse than before, thus fail to benefit the system output.
2) Entangled enhancement: The retraining of the model improves that model in its performance. However, due to the component entanglement, the improvement fail to benefit the system output.

For the degenerative retraining, the main cause might be biased choice of training and validation dataset, insufficient amount of data, or over-fitting caused by over-training. For the entangled enhancement, the cause is usually due to component entanglement.

b) self-defeating improvement: Self-defeating improvement is a representative example of entangled enhancement that can occur when an update to the upstream model does not result in the improvement in performance for the downstream model. For a system that has two-components, the entangled enhancement is exactly the same as self-defeating improvement, as there is only one downstream model, and the output of the downstream model is also the output of the system. The self-defeating improvement can affect the system state in two ways: 1. When the downstream model fails to achieve the threshold, but the upstream model achieves the threshold. In this condition, an update in the upstream model could possibly resulting in the worsening of the downstream model. 2. When both the upstream model and the downstream model fail to achieve the threshold. In this condition, if an update in the upstream model results in a self-defeating improvement, the downstream model may not reach the threshold. We examine these two cases in our state-space models and analysis in Section IV.

IV. Availability Modeling

A. Motivation

Considering the impact of imperfect retraining in an MLS, we aim to answer the question: **What can we do to maximize the MLS service availability?** We investigate the dynamic system's availability under the threat of the dataset shift and imperfect retraining in a two-component MLS. Availability is a system performance metric which provides insight into the coverage factor that an item or system will be available to be committed to a specified requirement [14]. In this study, we regard that the system is available when the downstream model's performance on the dataset X_{real} meets the required threshold τ_d. Otherwise, the system is regarded as unavailable. We leverage stochastic models to capture the state transitions of the MLS and quantitatively evaluate the service availability by the probability that MLS is in available states in the operation.

To maximize the service availability by effective retraining of ML models, we propose two model retraining policies, the progressive retraining policy and the conservative retraining policy. The progressive retraining policy reflects the strategy of actively tuning the system to keep the system updated, while the conservative retraining policy attempts retraining only when it is necessary. A comparison between these two policies is conduced through numerical analysis to analyze the effect of these policies on the service availability.

B. Progressive Retraining Policy

Under this policy, ML models are retrained from time to time to counter the potential performance degradation due to dataset shift. If both upstream model and downstream model satisfy the threshold, the policy performs no retraining is made.

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

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

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).

We model the state transitions of a two-component MLS under the proactive retraining policy by the CTMC as shown in Figure 2. We assume that the state transition times follow the exponential distributions, but the assumption can be relaxed by

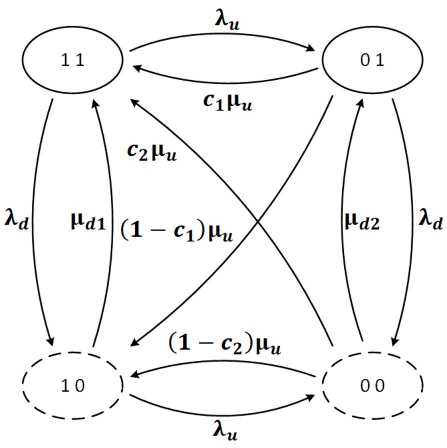


Fig. 2: A CTMC depicting the dynamic of MLS under progressive retraining policy. The failure rate due to dataset shift for u is λ_u, and for d is λ_d. The recovery rate for u is μ_u, and for d, if it is at the state (1,0), it is μ_{d1}, and if it is at the state(0,0), it is μ_{d2}. c_1 is the coverage factor that at state $(0,1)$, the recovery to u resulted in a transition to state $(1,1)$. c_2 is the coverage factor that at state $(0,0)$, the recovery to u resulted in a transition to state $(1,1)$.

extending the CTMC to semi-Markov process [8]. We denote the dataset shift transition rate for upstream model u is λ_u, and the dataset shift transition rate for downstream model d is λ_d. The successful recovery rate for upstream model u is μ_u. For downstream model d's retraining, if the upstream model u is above the threshold, the successful recovery rate is μ_{d1}, and if the upstream model u is below the threshold, the successful recovery rate is μ_{d2} ($> \mu_{d1}$), because recovering the performance of the downstream model is harder when the upstream model is deteriorated. c_1 is the coverage factor that at state (0,1), a successful retraining of the upstream model causes no entangled enhancement, thus not failing the downstream model. c_2 is the coverage factor that when the upstream model is retrained at state (0,0), both the downstream and upstream models satisfy the thresholds.

At state (1,1), no recovery should be observed. Due to the dataset shift, the system might experience compromise in either upstream model u or downstream model d. At state (0,1), the upstream model u fails to reach the threshold while downstream model d remains to be above the threshold. Therefore, only the successful retraining to the upstream model u can be observed. The retraining could be successful, leading a transition back to state (1,1), or it could trigger a entangled enhancement, which transits to state (1,0). At state (0,0), both upstream and downstream models fail to reach the threshold, and hence go under retraining attempts. However, retraining 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 entangled enhancement, or the downstream model only is imporved slightly, which is not enough for the state change. 2. The retraining is successful enough that the downstream model is also benefited from the improvement, so that it reaches to the threshold τ_d. At state (1,0), since only downstream model d is deteriorated, only successful retraining to the downstream model could be observed.

Because each state is reachable from other states, this CTMC is irreducible. For an irreducible CTMC, the state probabilities reach an asymptotic value as the time goes to infinity, and this asymptotic value is independent of the initial condition, and is called steady-state probability. We further denote the steady-state availability of the system as $\boldsymbol{\pi} = \{\pi_i\}$, $i \in \{(0,0),(0,1),(1,0),(1,1)\}$. $\boldsymbol{\pi}$ can be obtained by solving the system $\boldsymbol{\pi} \cdot Q = 0$ with $\boldsymbol{\pi} e^{\mathrm{T}} = 1$, where Q is the infinitesimal generator matrix of the CTMC, and e^{T} is the 4-dimension column vector whose elements are 1.

$$Q = \begin{bmatrix} -\lambda_u - \lambda_d & \lambda_u & \lambda_d & 0 \\ c_1\mu_u & -\mu_u - \lambda_d & (1-c_1)\mu_u & \lambda_d \\ \mu_{d1} & 0 & -\lambda_u - \mu_{d1} & \lambda_u \\ c_2\mu_u & \mu_{d2} & (1-c_2)\mu_u & -\mu_u - \mu_{d2} \end{bmatrix}. \tag{2}$$

Because the final output of the MLS is the downstream model's output, state $(1,1)$ and $(0,1)$ can be counted as available states. Thus, the service availability of the system under the progressive retraining policy is given by $A_p = \pi_{(1,1)} + \pi_{(0,1)}$.

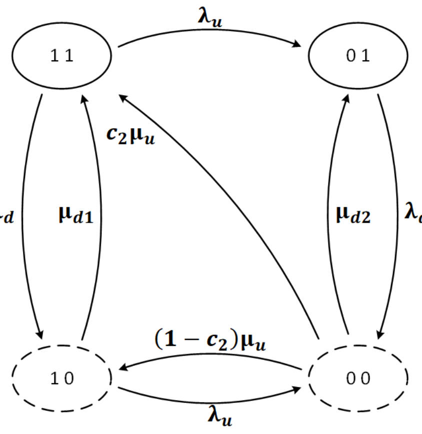


Fig. 3: A CTMC depicting the dynamic of MLS under conservative retraining policy. The failure rate due to dataset shift for u is λ_u, and for d is λ_d. The recovery rate for u is μ_u, and for d, if it is at the state (1,0), it is μ_{d1}, and if it is at the state(0,0), it is μ_{d2}. c_2 is the coverage factor that at state $(0,0)$, the recovery to u resulted in a transition to state $(1,1)$.

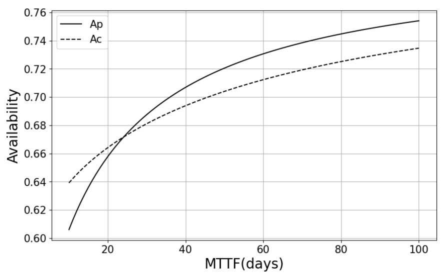
C. *Conservative Retraining Policy*

Next, we introduce the conservative retraining policy. Under this policy, ML models are retrained only when observing the deteriorated system performance (i.e., the performance of downstream model does not satisfy the required threshold τ_d).

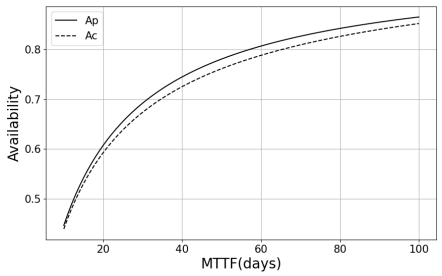
The two components MLS under the conservative retraining policy is modeled by the CTMC as shown in Fig. 3. At

TABLE I: Parameter Input

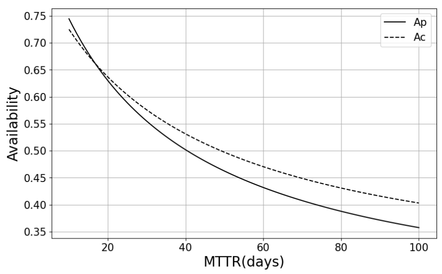
Parameter	Description	Value
$\frac{1}{\lambda_u}$	Mean time for upstream model performance to drop below threshold τ_u due to dataset shift	80 days
$\frac{1}{\lambda_d}$	Mean time for downstream model performance to drop below threshold τ_d due to dataset shift	40 days
$\frac{1}{\mu_u}$	Mean time for upstream model recovery	20 days
$\frac{1}{\mu_{d1}}$	Mean time for downstream model recovery when upstream model satisfies τ_u	10 days
$\frac{1}{\mu_{d2}}$	Mean time for downstream model recovery when upstream model fails to satisfy τ_u	100 days
c_1	The coverage factor of upstream model recovery not making downstream model unavailable	0.75
c_2	The coverage factor of upstream model recovery making both upstream model and downstream model available	0.5



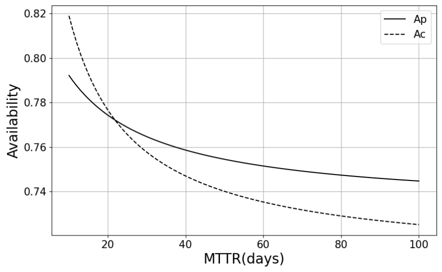
(a) A_p and A_c: λ_u



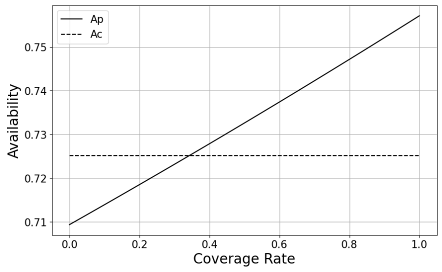
(b) A_p and A_c: λ_d



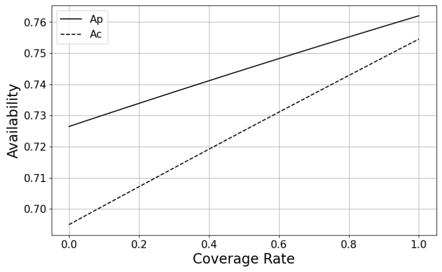
(c) A_p and A_c: μ_{d1}



(d) A_p and A_c: μ_{d2}



(e) A_p and A_c: c_1



(f) A_p and A_c: c_2

Fig. 4: Service Availability regarding to different parameter changes

state (1,1) and (0,1), no retraining is attempted because the performance of the downstream model satisfies the threshold. At State (1,0) and (0,0), both upstream and downstream models are being retrained. Compared to the system in Fig. 2, the transition from (0,1) to (1,1) is removed. However at the same time, the transition from (0,1) to (1,0), which is the entangle enhancement that causes the system to fail, is also removed, which might benefit the system.

As the CTMC is irreducible, we can compute the steady-state probabilities through the infinitesimal generator:

$$Q = \begin{bmatrix} -\lambda_u - \lambda_d & \lambda_u & \lambda_d & 0 \\ 0 & -\lambda_d & 0 & \lambda_d \\ \mu_{d1} & 0 & -\lambda_u - \mu_{d1} & \lambda_u \\ c_2\mu_u & \mu_{d2} & (1-c_2)\mu_u & -\mu_u - \mu_{d2} \end{bmatrix}. \quad (3)$$

The service availability under the conservative retraining policy can again be derived as $A_p = \pi_{(1,1)} + \pi_{(0,1)}$.

V. Numerical Analysis

A. Parameters Assignment

Table I presents the input parameter values employed in the analysis. Due to a lack of literature addressing the temporal distribution of dataset shift in real-world scenarios, these values are selected based on assumptions for an MLS operation. Specifically, the mean time to upstream model failure attributable to dataset shift is set to 80 days, and the mean time to upstream model recovery is set to 20 days. For the downstream model, the mean time to failure is set to 20 days, and the mean time for downstream model recovery, triggered when the upstream model reaches the threshold τ_u, is set to 10 days. The mean time for downstream model recovery when the upstream model fails to reach τ_u is configured as 100 days, reflecting the anticipation that such events are unlikely to occur. The coverage factor of an upstream model recovery leading to no downstream model failure when downstream model is already available (denoted as c_1) is set to 0.75. Similarly, the converge factor of an upstream model recovery resulting in the downstream model reaching τ_d and upstream model reaching τ_u (denoted as c_2) is set to 0.5.

B. Sensitivity Analysis

Tables II shows the sensitivities of transition rates and coverage factors influencing A_p and A_c. The two most significant transition rates influencing service availability of the

TABLE II: Scaled Sensitivity for A_p and A_c

Parameter θ	$SS_\theta(A_p)$	$SS_\theta(A_c)$
λ_u	-0.0602	-0.0602
λ_d	-0.229	-0.233
μ_u	0.0802	0.122
μ_{d1}	0.194	0.149
μ_{d2}	0.0147	0.0218
c_1	0.0165	-
c_2	0.0238	0.0411

system is λ_d, followed by μ_{d1}. Obviously, the recovery rate and coverage factor positively impacts service availability, whereas an increase in the failure rate negatively affects service availability.

We plot A_p and A_c by varying the values of individual parameters as shown in Fig. 4. The trends exhibited by the lines in the figures mostly align with the parameters shown in the tables. Fig. 4 (a) and (b) show the sensitivity to the dataset transition rate. In Fig. 4 (b), there is a changing point where the service availability achieved by the progressive retraining policy overcomes the conservative retraining policy. This phenomenon can also be observed in Fig. 4 (c) and (d), which present the sensitivity to the downstream model recovery rate at (1,0) and (0,0), respectively. As shown in the figures, when μ_{d1} becomes large enough, service availability under the progressive retraining policy exceeds the one under the conservative retraining policy, while μ_{d2} shows the completely opposite trend, as when it becomes large enough, service availability under the conservative retraining policy starts to win over the progressive retraining policy. This can be attributed to the fact that under the progressive retraining policy, system is more likely to experience deterioration due to the entangled enhancement represented by the transition from (0,1) to (1,0). μ_{d1}, as the recovery rate from (1,0) to (1,1), can possibly revert the bad influence of entangled enhancement, thus compensating the progressive retraining policy more. On the other hand, μ_{d2} is the recovery rate from (0,0) to (0,1), which would benefit service availability under the conservative retraining policy more, as it is more likely to visit state (0,0) as a failure state. Fig. 4 (e) shows the sensitivity to the coverage factor for transition (0,1) to (1,1). When c_1 is sufficiently small, the progressive retraining policy becomes worse than the conservative policy due to potential negative impacts of the entangled enhancement. In summary, the efficacy of a system's retraining policies in terms of service availability is notably influenced by λ_u, μ_{d1}, μ_{d2} and c_1, which play pivotal roles in determining the policy that yields a more available system.

VI. Conclusion

This study addresses the challenges encountered by MLSs in the presence of dataset shifts, introducing the concept of imperfect retraining, with a specific focus on entangled enhancement. We proposed and evaluated two maintenance policies, namely the progressive retraining policy and the conservative retraining policy, to counteract the performance deterioration due to dataset shift. CTMCs are employed to capture the dynamics of dataset shift and entangled enhancement within a two-model MLS. Sensitivity analysis is then applied to assess the effectiveness of each policy under different circumstances. The findings indicate that, concerning service availability, both policies exhibit advantages under distinct circumstances. Regardless of the policy, the failure rate of the downstream model is of greatest significance, and hence reducing the failure rate is the primary consideration for improving availability.

References

[1] 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.

[2] 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.

[3] 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.

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

[5] K. Wang, S. Zhou, C.A. Fu, J.X. Yu, F. Jeffrey and X. Yu, "Mining changes of classification by correspondence tracing," in Proceedings of the 2003 SIAM International Conference on Data Mining, pp. 95–106, 2003.

[6] Y. Yang, X. Wu, and X. Zhu, "Conceptual equivalence for contrast mining in classification learning," Data & Knowledge Engineering, vol. 67, no. 3, pp. 413–429, 2008.

[7] D.A. Cieslak and N.V. Chawla, "A framework for monitoring classifiers' performance: when and why failure occurs?" in Knowledge and Information Systems, vol. 18, no. 1, pp. 83–108, 2009.

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

[9] 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.

[10] 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.

[11] F. Machida, J. Xiang, K. Tadano and Y. Maeno, "Lifetime Extension of Software Execution Subject to Aging," in IEEE Transactions on Reliability, vol. 66, no. 1, pp. 123-134, 2017.

[12] 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.

[13] 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.

[14] D. J. Hurst, "Operational availability modeling for risk and impact analysis," in Annual Reliability and Maintainability Symposium 1995 Proceedings. IEEE, pp. 391–396, 1995.

[15] G. C. Cawley and N. L. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," The Journal of Machine Learning Research, vol. 11, pp. 2079–2107, 2010.

[16] 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.