RegTrieve: Reducing System-Level Regression Errors for Machine Learning Systems via Retrieval-Enhanced Ensemble

JUNMING CAO, Fudan University, China
XUWEN XIANG, Fudan University, China
MINGFEI CHENG, Singapore Management University, Singapore
BIHUAN CHEN*, Fudan University, China
XINYAN WANG, Fudan University, China
YOU LU, Fudan University, China
CHAOFENG SHA, Fudan University, China
XIAOFEI XIE, Singapore Management University, Singapore
XIN PENG, Fudan University, China

Multiple machine learning (ML) models are often incorporated into real-world ML systems. However, updating an individual model in these ML systems frequently results in regression errors, where the new model performs worse than the old model for some inputs. While model-level regression errors have been widely studied, little is known about how regression errors propagate at system level. To address this gap, we propose RegTrieve, a novel retrieval-enhanced ensemble approach to reduce regression errors at both model and system level. Our evaluation across various model update scenarios shows that RegTrieve reduces system-level regression errors with almost no impact on system accuracy, outperforming all baselines by 20.43% on average.

CCS Concepts: • **Software and its engineering → Software evolution**.

Additional Key Words and Phrases: Regression Error, Ensemble Model, Spoken QA System

ACM Reference Format:
Junming Cao, Xuwen Xiang, Mingfei Cheng, Bihuan Chen, Xinyan Wang, You Lu, Chaofeng Sha, Xiaofei Xie, and Xin Peng. 2025. RegTrieve: Reducing System-Level Regression Errors for Machine Learning Systems via Retrieval-Enhanced Ensemble. *Proc. ACM Softw. Eng.* 2, FSE, Article FSE088 (July 2025), 23 pages. https://doi.org/10.1145/3729358

1 Introduction

Machine learning (ML) models are widely deployed in various real-world applications, e.g., question-answering systems [12, 36] and autonomous driving systems [33]. Such complex ML systems often

*Bihuan Chen is the corresponding author.

Authors' Contact Information: Junming Cao, School of Computer Science, Fudan University, Shanghai, China, 21110240004@m.fudan.edu.cn; Xuwen Xiang, School of Computer Science, Fudan University, Shanghai, China, 21302010055@m.fudan.edu.cn; Mingfei Cheng, Singapore Management University, Singapore, Singapore, mfcheng.2022@phdcs.smu.edu.sg; Bihuan Chen, School of Computer Science, Fudan University, Shanghai, China, bhchen@fudan.edu.cn; Xinyan Wang, School of Computer Science, Fudan University, Shanghai, China, 22302010044@m.fudan.edu.cn; You Lu, School of Computer Science, Fudan University, Shanghai, China, ylu24@m.fudan.edu.cn; Chaofeng Sha, School of Computer Science, Fudan University, Shanghai, China, cfsha@fudan.edu.cn; Xiaofei Xie, Singapore Management University, Singapore, Singapore, xfxie@smu.edu.sg; Xin Peng, School of Computer Science, Fudan University, Shanghai, China, pengxin@fudan.edu.cn.

ACM 2994-970X/2025/7-ARTFSE088
https://doi.org/10.1145/3729358

integrate multiple models. As these systems evolve, updating an individual model may lead to *regression errors*, i.e., the new version of a model performs worse than the old version for some inputs. Such regression errors can have a negative impact on ML systems. For example, in ConvLab [65], a widely-used task-oriented dialogue system, replacing the dialogue state tracking component from SUMBT [21] to TRADE [54] increases the slot accuracy of this component from 96.44% to 96.92%, but decreases the overall system success rate from 27.8% to 22.4% [40]. This indicates that the number of introduced system-level regression errors exceeds the newly added successful tasks. In July 2021, an update to the *face unlock* feature of the Galaxy S10 5G led to authentication failures for users, which harmed Samsung's reputation and resulted in economic losses [49].

Regression errors at a single model level [10, 23, 50, 57, 58] have recently been widely investigated. However, there is limited understanding of how regression errors propagate at the system level, where multiple models are integrated. Moreover, it is difficult for existing training-based [10, 57, 58] and ensemble-based [23, 50] model-level regression error reduction approaches to extend to the system level, as it is challenging to efficiently gather information from downstream models when updating a model. For example, Li et al. [23] propose an uncertainty-based ensemble approach to mitigate model-level regression errors. However, this approach is difficult to apply to the system level because obtaining the uncertainty of downstream models during inference is time-consuming.

To address this problem, we first extend the formulation of regression errors from model-level to system-level, focusing on the impact of model updates within ML systems containing multiple ML models. Then, inspired by retrieval-augmented generation (RAG) techniques [17, 20, 41] used to enhance large language models, we find that retrieval methods can efficiently gather information from the overall system, thereby aiding ensemble-based approaches from a system perspective. Ensemble-based approaches reduce regression errors by combining the prediction results of the new model and the old model with ensemble weight. The main challenge is to find the appropriate ensemble weights for the old and new models for a specific testing sample. Therefore, we propose a retrieval-enhanced ensemble approach, RegTrieve, to reduce both model-level and system-level regression errors effectively and efficiently. RegTrieve estimates the accuracy of the old and new models on a given testing sample by retrieving similar training samples and using the losses from these training samples to determine the ensemble weights for the two models. The losses can represent system-level losses by performing a forward pass of the entire system. As a result, RegTrieve effectively reduces system-level regression errors. After the datastore for retrieval is built offline, RegTrieve's inference latency remains low, as it only adds the slight overhead of retrieving the nearest training samples.

We first conduct comprehensive experiments to demonstrate the prevalence of system-level regression errors. We find that system-level regression errors often arise independently of model-level regressions. Specifically, there are only lower than 20% system-level regression errors are caused by model-level regression errors. These results motivate the need for specialized approaches to reduce system-level regression errors. Then, we conduct extensive experiments to evaluate the effectiveness and efficiency of RegTrieve across various model update scenarios in the spoken QA system. The results demonstrate that RegTrieve can reduce system-level regression errors by an average of 2.21% (via model losses) and 2.40% (via system losses), with little or even no loss in system F1 score (i.e., less than 0.5%), which outperforms all baselines by 20.43% on average. Moreover, RegTrieve requires only about 20% of the inference time required by uncertainty-based approaches. Finally, compared with baseline approaches, RegTrieve shows a promising generalization capability by reducing 14.06% more system-level regression errors in the multi-sensor fusion perception system.

In summary, this work makes the following main contributions.

- We extend the regression error problem from model-level to system-level and demonstrate that system-level regression errors cannot always be inferred from model-level regression errors.

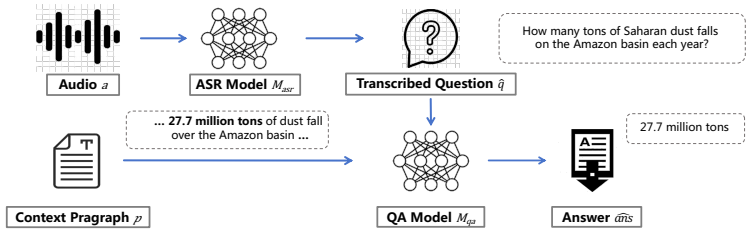


Fig. 1. An illustration of the spoken QA system

- We propose RegTrieve, a novel retrieval-enhanced ensemble approach to reduce both model-level and system-level regression errors without compromising overall system accuracy.
- We evaluate our approach through extensive experiments, demonstrating significant improvements in reducing regression errors compared to baseline approaches.

2 Problem Formulation and Motivation

We first introduce a spoken question answering (QA) system, which will be used as a running example. Then, without loss of generality, we formulate the model-level and system-level regression errors for the spoken QA system, which could be easily extended to other ML systems. Finally, we present why system-level regression errors matter with examples.

2.1 The Running Example of Spoken QA System

As Fig. 1 shows, the spoken QA system consists of an automatic speech recognition (ASR) model and a downstream question answering (QA) model. Given an audio file a as an input, the ASR model transcribes it into a question text $\hat{q}$. The QA model generates an answer $\hat{ans}$ based on $\hat{q}$ and a context paragraph p which is prepared by the QA dataset and should be retrieved from corpus in real world. We apply word error rate (WER) and F1 score to measure the accuracy of ASR model and QA model, respectively, which are widely used in the literature [22, 36, 56].

Generally, the ASR model transcribes the audio file in a self regression fashion by Eq. 1,

$$P(x_k \mid a) = P(x_k \mid a, x_{<k}) = softmax(z_k) \tag{1}$$

where $P(x_k \mid a)$ is the posterior probability of k_{th} token over vocabulary V conditioned by the audio file a. It is computed from the *softmax* of logits z_k produced by the ASR model. With a greedy decoding strategy, every time the model selects the token $\hat{x_k}$ with the maximum probability. Once the end of sentence (EOS) token is generated, the transcribed process finishes. Given a transcribed question text $\hat{q} = (\hat{x_1}, \hat{x_2}, \ldots, \hat{x_n})$ and a context paragraph p, the extractive QA model [19, 25] predicts the start and end position of the answer span within the context paragraph, i.e., $P(s \mid \hat{q}, p)$ and $P(e \mid \hat{q}, p)$, and then selects the answer $\hat{ans}$ with the maximum probability.

2.2 Model-Level Regression Error Formulation

Given an ASR testing dataset $\mathcal{D}_{test}$ of N samples with groud-truth transcribed questions, i.e., $\mathcal{D}_{test} = \{(a_1, q_1), (a_2, q_2), \ldots, (a_N, q_N)\}$, the WER of an ASR model M on a sample (a_i, q_i) is defined by Eq. 2,

$$wer(q_i, \hat{q_i}) = \frac{S_i + D_i + I_i}{|q_i|} \tag{2}$$

where $|q_i|$ denotes the length of question q_i, and S_i, D_i and I_i respectively denote the number of substitutions, deletions and insertions to transform $\hat{q_i}$ into q_i. Usually, we update an old model M^{old} to a new model M^{new} for accuracy improvement, support of more label classes, model compression, etc. There will be *positive flips* (PFs) that M^{new} corrects the mispredictions of M^{old}, and *negative flips*

(NFs), i.e., *regression errors*, that M^{new} misclassifies but M^{old} classifies correctly. The model accuracy improvement or degradation originates from the difference between PFs and NFs. Formally, the regression error rate at model level is defined by Eq. 3,

$$Reg(M^{old}, M^{new}) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left(wer(q_i, \hat{q_i}^{new}) - wer(q_i, \hat{q_i}^{old}) \geq \tau_m\right) \tag{3}$$

where $\mathbf{1}$ is an identifier function to indicate whether WER increases more than a threshold τ_m after model update. We define regression error reduction as a constrained optimization problem by Eq. 4,

$$\underset{M^{'}}{\text{minimize}} \quad Reg(M^{old}, M^{'}), \quad \text{subject to} \quad WER(M^{'}) \leq WER(M^{new}) \tag{4}$$

where $M^{'}$ denotes the improved model produced by regression error reduction techniques (e.g., model ensemble [23] and knowledge distillation [57, 58]), and $WER(M^{'})$ ad $WER(M^{new})$ respectively denote the average WER of the model $M^{'}$ and M^{new} over the whole testing dataset. It is a kind of *Pareto Improvement* [4] because regression errors are reduced without any sacrifice of accuracy.

2.3 System-Level Regression Error Formulation

A spoken QA system S consists of an ASR model M_{asr} and a QA model M_{qa}, i.e., $S = \{M_{asr}, M_{qa}\}$. We measure the accuracy of the overall system with F1 score of M_{qa}, which is calculated based on the overlap between the predicted answer and the ground-truth answer. Generally, the ASR model and QA model should be updated individually to evaluate their impact on the whole system fairly. Here, we consider the system update in the ASR model update scenario, while keeping the QA model not updated, i.e., $S^{new} = \{M_{asr}^{new}, M_{qa}^{old}\}$ and $S^{old} = \{M_{asr}^{old}, M_{qa}^{old}\}$. There are also PFs and NFs (i.e., *regression errors*) at system level. Formally, the regression error rate at system level is defined by Eq. 5,

$$Reg(S^{old}, S^{new}) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\left(F1(ans_i, \hat{ans_i}^{old}) - F1(ans_i, \hat{ans_i}^{new}) \geq \tau_s\right) \tag{5}$$

where $F1(ans_i, \hat{ans_i}^{old})$ denotes the F1 score of word set in predicted answer $\hat{ans_i}^{old}$ compared with that in ground-truth answer ans_i, and $\mathbf{1}$ denotes an identifier function to indicate whether F1 score decreases more than a threshold τ_s after system update. We formalize the system-level regression error reduction problem by Eq. 6,

$$\underset{S^{'}}{\text{minimize}} \quad Reg(S^{old}, S^{'}), \quad \text{subject to} \quad F1(S^{'}) \geq F1(S^{new}) \tag{6}$$

where $S^{'}$ denotes the improved system produced by regression error reduction techniques, and $F1(S^{'})$ and $F1(S^{new})$ denote the average F1 score of the system $S^{'}$ and S^{new} on the whole testing dataset.

2.4 Why System-Level Regression Errors Matter

System-level updates always happen with model-level updates as we update one or several models to update the system. Then why should we consider regression errors at system level? Does the reduction of regression errors at model level also reduce those at system level? Li et al. [22] and Wu et al. [56] found that ASR errors have huge negative impact on the accuracy of downstream QA model. Inspired by them, Su et al. [39] and You et al. [61] proposed to use contextualized word representations and knowledge distillation to improve the overall system accuracy. However, system-level regression errors in spoken QA system are still unexplored.

Srivastava et al. [38] and Zhu et al. [3] show that the accuracy improvement of a single model does not indicate the improvement of the whole ML system, denoted as Eq. 7 in spoke QA system.

$$WER(M^{new}) \leq WER(M^{old}) \implies F1(S^{new}) \geq F1(S^{old}) \tag{7}$$

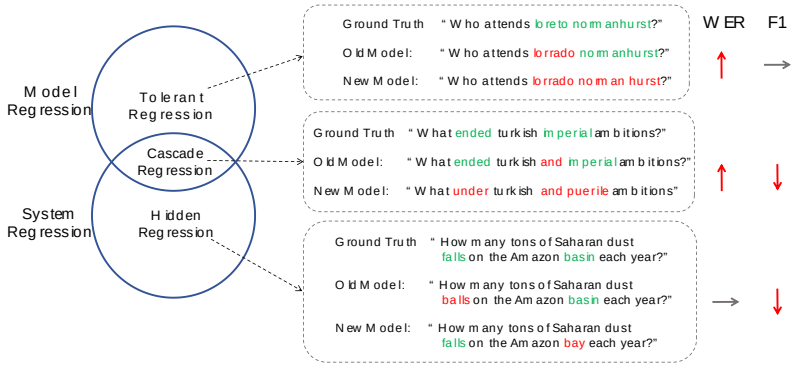


Fig. 2. The relation between model-level and system-level regression errors

The main reason for Eq. 7 to hold is the data distribution mismatch between evaluations at model and system level [3, 38]. For example, the new ASR model may perform better for clean audios but worse for noisy audios, while the QA model performs good enough for clean audios. Thus, the QA model cannot benefit from the improvement of the new ASR model but be harmed by its regression errors.

By swapping the old and new model, the reverse version of Eq. 7 holds, meaning that the accuracy decrease of a single model does not necessarily result in a reduction in the overall system accuracy. However, Eq. 7 only presents the overall accuracy relation, and it does not present the relation between regression error at model and system level for a specific testing sample. Therefore, we extend the reverse version of Eq. 7 from the overall accuracy to a single testing sample level, i.e., *the accuracy reduction of a single testing sample at model level does not indicate that at system level* by Eq. 8.

$$wer(q_i, \hat{q}_i{}^{new}) \geq wer(q_i, \hat{q}_i{}^{old}) \implies F1(ans_i, a\hat{n}s_i{}^{new}) \leq F1(ans_i, a\hat{n}s_i{}^{old}) \tag{8}$$

Eq. 8 implies that model-level regression errors and system-level regression errors do not necessarily occur together. As shown in Fig. 2, they represent two intersecting sets with three distinct parts. Fig. 2 also provides three examples to illustrate the three parts. Each example includes the transcription result from the ASR model after an update, and indicates whether it leads to a model-level regression (i.e., increase in WER) or a system-level regression (i.e., decrease in F1 score).

- **Tolerant Regression:** model-level regression errors that do not cause the system to fail. As the first example in Fig. 2 shows, given an audio input where the ground truth transcription is "Who attends Loreto Normanhurst?", the old ASR model incorrectly recognized "Loreto" as "Lorrado." The new ASR model further introduced an error by recognizing "Normanhurst" as "Norman Hurst." Although WER increased for the new model, the additional error was not critical and did not affect the QA model's ability to understand the question. Therefore, the QA model's prediction remained correct, and no system-level regression occurred.
- **Cascade Regression:** model-level regression errors that cause the system to fail. As the second example in Fig. 2 shows, the new ASR model introduced errors by recognizing "ended" as "under" and "imperial" as "puerile". These new errors significantly altered the meaning of the question, preventing the QA model from answering correctly, resulting in a system-level regression error.
- **Hidden Regression:** system-level regression errors that do not occur at model level. As the third example in Fig 2 shows, the old ASR model incorrectly recognized "falls" as "balls". Luckily, the new ASR model correctly recognized "falls", but it incorrectly recognized "basin" as "bay". These two models all misrecognized one word, meaning that this is not a regression error for the ASR model. However, the downstream QA model could correctly answer the input from the

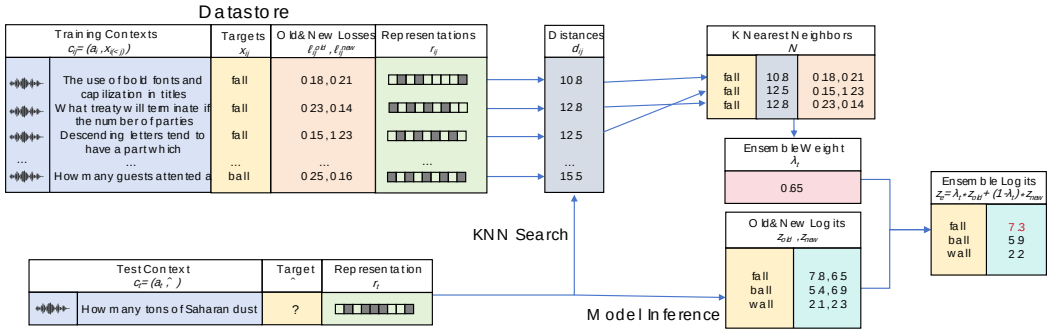


Fig. 3. An illustrative example of RegTrieve's inference

old ASR model, but not for that from the new ASR model. The reason is that "basin" is more important than "falls" for the QA model to understand and answer the question correctly.

We will demonstrate in Sec. 5.1 that there are many tolerant and hidden regression errors in practice. Thus, regression errors should be reduced at system level, apart from model level.

3 Our Approach

We propose a retrieval-enhanced ensemble approach to reduce regression errors at both model and system level. We develop it on ensemble-based approaches, as they are training-free and reduce model-level regression errors effectively while preserving the original accuracy [23].

3.1 The Framework of RegTrieve

Generally, an ensemble ASR model is the weighted average of the old model and new model. The ensemble logits is computed by $z_e = \lambda * z_{old} + (1 - \lambda) * z_{new}$ first, where λ is the ensemble weight ranging from 0 to 1. Then, as Eq. 1 shows, the *softmax* function is applied on logits z_e to obtain the normalized probability distribution p_e over the vocabulary V. However, it is hard to decide the most appropriate ensemble weight λ to reduce the regression errors without accuracy degradation, as the appropriate λ differs for different testing samples. The intuition of ensemble-based approaches is that the new model improves the accuracy of the old model for some testing samples but harms it for other testing samples. Inspired from this, Li et al. [23] estimate λ based on the uncertainty of the old and new models for a given testing sample.

The intuition of our retrieval-enhanced ensemble approach is to *utilize similar training samples to surrogate the accuracy of the old model and new model on a testing sample*, which could be used to decide the proper ensemble weight for the specific testing sample. This approach is training-free and only need model forward passes to create the indexed datastore. As the ASR model transcribes token by token during the inference time, we need to create token-level datastore first, then retrieve neighbors and ensemble model predictions token by token. Fig. 3 presents an example to illustrate RegTrieve's inference process. Given a testing audio file a_t and a partially predicted question $\hat{q}$, we need to predict the next token. First, based on the vector representation r_t of the current context, K nearest neighbors (KNN) are retrieved from the datastore. Then, the ensemble weight λ_t is calculated by performing a weighted average using the neighbors' old loss, new loss, and distance. Finally, the ensemble logits z_e are obtained by combining the old and new models' logits with λ_t. We formulate how to construct the datastore and perform inference in RegTrieve as follows.

Datastore. Given an ASR training dataset $\mathcal{D}_{train} = \{(a_1, q_1, p_1, ans_1), \dots\}$, we create the datastore $\mathcal{H}_{train}$ consisting of *token samples* with Alg. 1. In total, there are $|\mathcal{D}_{train}| * L$ token samples in $\mathcal{H}_{train}$, where L is the average token sequence length in $\mathcal{D}_{train}$. For the j_{th} token of the i_{th} audio in

Algorithm 1 Construction of the Datastore

1: **Input:** a training dataset $\mathcal{D}_{train}$, an old spoken QA system $S^{old} = \{M^{old}_{asr}, M^{old}_{qa}\}$, a new spoken QA system $S^{new} = \{M^{new}_{asr}, M^{old}_{qa}\}$
2: **Output:** a datastore $\mathcal{H}_{train}$ that consists of *token samples* as key-value pairs
3: $\mathcal{H}_{train} = \emptyset$.
4: **for** (a_i, q_i, p_i, ans_i) in $\mathcal{D}_{train}$ **do**
5: *// Iterate over every token for a given ground-truth question text q_i*
6: **for** t_j in q_i **do**
7: $c_{ij} = (a_i, x_{i(<j)})$, where c_{ij} is the context of j_{th} token in the i_{th} training sample from $\mathcal{D}_{train}$
8: $r_{ij} = \mathcal{R}(M^{new}_{asr}, c_{ij})$, where $\mathcal{R}$ gets the intermediate representation of M^{new}_{asr} for context c_{ij}
9: $\ell^{old}_{ij} = \mathcal{L}(M^{old}_{asr}, c_{ij}, x_{ij})$, $\ell^{new}_{ij} = \mathcal{L}(M^{new}_{asr}, c_{ij}, x_{ij})$, where x_{ij} is the ground-truth token for current context, and $\mathcal{L}$ is the model-level loss function
10: $g^{old}_{ij} = \mathcal{G}(S^{old}, c_{ij}, x_{ij})$, $g^{new}_{ij} = \mathcal{G}(S^{new}, c_{ij}, x_{ij})$, where $\mathcal{G}$ is the system-level loss function
11: $\mathcal{H}_{train} = \mathcal{H}_{train} \cup \{(r_{ij}, (\ell^{old}_{ij}, \ell^{new}_{ij}, g^{old}_{ij}, g^{new}_{ij}))\}$
12: **end for**
13: **end for**
14: **return** $\mathcal{H}_{train}$

$\mathcal{D}_{train}$, the key of its corresponding token sample in $\mathcal{H}_{train}$ is r_{ij}, while the value is model-level and system-level losses (see Sec. 3.2 and 3.3). r_{ij} is created from the intermediate representation of the new model (Line 8), e.g., the last hidden states in transformer-based architecture. We utilize the representation from the new model, instead of the old model, because it is supposed to be more accurate. Each token sample is accompanied with model-level losses l_{ij} and system-level losses g_{ij} of the old and new models (Line 9–10). They are used during inference to estimate ensemble weight.

Inference. Given an ASR testing dataset $\mathcal{D}_{test} = \{(a_1, p_1), \dots\}$, we transcribe it with Alg. 2, and then apply the downstream QA model to predict the answer for transcribed question $\hat{q}$ according to context paragraph p. Essentially, the logits from the old and new model are combined token by token (Line 14-15), according to the ensemble weight that is computed from similar neighbors' losses in $\mathcal{H}_{train}$. At Line 11–13, the model-level ensemble weight λ^m_t is calculated based on model-level losses and ensemble function. Similarly, we replace model-level losses and ensemble function with system-level losses and ensemble function to obtain system-level ensemble weight λ^s_t.

We have described the overall framework of our retrieval-enhanced ensemble approach. We will explain the model-level and system-level regression error reduction in detail. They share the same framework as presented before, but have different loss functions and loss ensemble functions.

3.2 Model-Level Regression Error Reduction

We first introduce how to reduce model-level regression errors with model-level losses and ensemble function. During the datastore construction, ℓ^{old}_{ij} and ℓ^{new}_{ij} at Line 9 in Alg. 1 are calculated by Eq. 9,

$$\ell_{ij} = -\log P(x_{ij} \mid c_{ij}) \tag{9}$$

where x_{ij} denotes the ground-truth token given the context $c_{ij} = (a_i, x_{i(<j)})$. Once the estimated old loss ℓ^{old}_t and new loss ℓ^{new}_t for the next token prediction are obtained at Line 11 in Alg. 2, we define the ensemble function $\mathcal{E}_m$ to compute the ensemble weight λ^m_t according to them. In principle, the larger $\ell^{diff}_t = \ell^{new}_t - \ell^{old}_t$ is, the larger λ^m_t should be, because the new model may perform worse than the old model to predict the specific next token. We have analyzed $\ell^{diff}_{ij} = \ell^{new}_{ij} - \ell^{old}_{ij}$ for all token samples in $\mathcal{H}_{train}$ in model update scenarios described in our evaluation setups (see Sec. 4), and found that they roughly follow the probability density distribution (PDF) of normal distribution. Therefore, it is natural to apply the cumulative distribution function (CDF) of normal distribution to map the current ℓ^{diff}_t into λ^m_t which ranges from 0 to 1.

Algorithm 2 Transcription with Retrieval-Enhanced Ensemble Approach

Input: a testing dataset $\mathcal{D}_{test}$, a datastore $\mathcal{H}_{train}$, an old spoken QA system $S^{old} = \{M_{asr}^{old}, M_{qa}^{old}\}$, a new spoken QA system $S^{new} = \{M_{asr}^{new}, M_{qa}^{old}\}$
Output: transcribed question set Q_e for testing samples in the testing dataset
$Q_e = \emptyset$
for (a_i, p_i) in $\mathcal{D}_{test}$ **do**
  $\hat{q} = [\,]$
  while True **do**
    $c_t = (a_i, \hat{q})$, where c_t is the current context to predict the next *token*. It consists of the audio a_i and partially transcribed question $\hat{q}$
    $r_t = \mathcal{R}(M_{asr}^{new}, c_t)$, where $\mathcal{R}$ gets the intermediate representation of M_{asr}^{new} for context c_t
    $N = Retrieve(r_t, \mathcal{H}_{train})$, where N is a set of the retrieved nearest neighbors with a size of K
    $w_j = softmax(\frac{d_j - \mu_t}{\sigma_t})$, where d_j is the L2 distance between the representation of the j_{th} neighbor r_j and that of context r_t, and μ_t and σ_t are the mean and standard deviation of distances in N
    $\ell_t^{old} = \sum_{j=1}^{K} w_j \ell_j^{old}$, $\ell_t^{new} = \sum_{j=1}^{K} w_j \ell_j^{new}$, where ℓ_t^{old} and ℓ_t^{new} are the estimated model-level loss of the old and new model for the next token prediction
    // Apply λ_t^s and $\mathcal{E}_s$ instead to reduce system-level regression errors
    $\lambda_t^m = \mathcal{E}_m(\ell_t^{old}, \ell_t^{new})$, where $\mathcal{E}_m$ is model-level ensemble function to obtain the ensemble weight
    $z_{old} = M_{asr}^{old}(c_t)$, $z_{new} = M_{asr}^{new}(c_t)$, where z_{old} and z_{new} are logits generated from M_{asr}^{old} and M_{asr}^{new}
    $z_e = \lambda_t^m * z_{old} + (1 - \lambda_t^m) * z_{new}$, $P(t \mid c_t) = softmax(z_e)$
    $\hat{x} = \arg\max_x P(x \mid c_t)$, where $\hat{x}$ is the predicted token by our ensemble approach
    $\hat{q} = [\hat{q}, \hat{x}]$
    // Finish when the end of text token is generated.
    if $\hat{x}$ == <eos> **then**
      break
    end if
  end while
  $Q_e = Q_e \cup \{\hat{q}\}$
end for
return Q_e

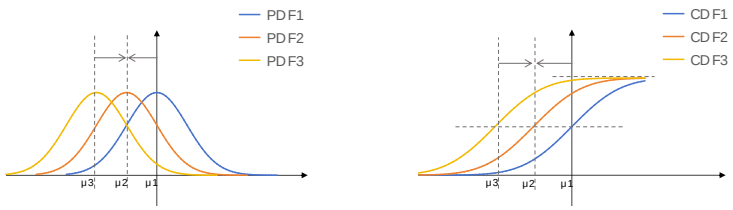


Fig. 4. An illustration of PDFs and CDFs of model-level loss difference

Specifically, we implement the model-level loss ensemble function $\mathcal{E}_m$ by Eq. 10,

$$\lambda_t^m = \mathcal{E}_m(\ell_t^{diff}) = \Phi\left(\frac{\ell_t^{diff} - \mu^m - \gamma^m * \sigma^m}{\sigma^m}\right) \tag{10}$$

where Φ is the CDF of a standard normal distribution with a mean of 0 and a standard deviation of 1. μ^m and σ^m respectively denote the mean and standard deviation of model-level loss difference ℓ_{ij}^{diff} of all token samples in $\mathcal{H}_{train}$. ℓ_t^{diff} is transformed into a random variable following the standard normal distribution, and then mapped to the ensemble weight using Φ. γ^m is a hyper-parameter, which is used to shift the PDF and CDF of the normal distribution for the following two reasons.

Regression Error Reduction. Consider a simple scenario where the accuracy of the new model is similar to that of the old model. The PDF and CDF of the loss difference distribution are PDF_1 and CDF_1, as illustrated in Fig. 4. If γ^m is set to 0 in Eq. 10, i.e., the PDF is not shifted, λ_t^m scatters

Algorithm 3 Approximation of System-Level Losses

1: **Input:** a training dataset $\mathcal{D}_{train}$, a datastore $\mathcal{H}_{train}$, a spoken QA system $S = \{M_{asr}, M_{qa}\}$
2: **Output:** the estimated system-level losses set G for token samples in $\mathcal{H}_{train}$
3: *// Step 1: Calculate system-level losses of training samples in $\mathcal{D}_{train}$*
4: **for** (a_i, q_i, p_i, ans_i) in $\mathcal{D}_{train}$ **do**
5: $\quad \hat{q}_i \leftarrow M_{asr}(a_i)$
6: $\quad \hat{ans}_i \leftarrow M_{qa}(\hat{q}_i, p_i)$
7: $\quad \hat{g}_i = -\log P(s \mid \hat{q}_i, p_i) - \log P(e \mid \hat{q}_i, p_i)$, where s and e respectively denote the start and end position of the ground-truth answer ans_i in the context paragraph p_i
8: **end for**
9: *//Step 2: Approximate system-level losses of token samples in $\mathcal{H}_{train}$*
10: $G = \emptyset$
11: **for** each token sample in $\mathcal{H}_{train}$ **do**
12: $\quad g_{ij} = \hat{g}_i$
13: $\quad G = G \cup g_{ij}$
14: **end for**
15: **return** G

around 0.5 when ℓ_t^{diff} is 0. However, when the accuracy of the old model is comparable to that of the new model for the current token prediction, we prefer an ensemble weight greater than 0.5, as the new model could not introduce significant accuracy improvement and may cause regression errors. As a result, the logits z_{old} from the old model is preferred to reduce regression errors. To achieve this goal, γ^m should be negative to shift the CDF to the left, as illustrated by CDF_2 in Fig. 4. In this way, the new λ_t^m will be larger than 0.5 on CDF_2 when ℓ_t^{diff} is 0.

Accuracy Preservation. Consider another scenario where the new model improves the accuracy significantly compared with the old model. The PDF and CDF of the loss difference distribution are PDF_3 and CDF_3, as illustrated in Fig. 4. If γ^m is set to 0 in Eq. 10, i.e., the CDF is not shifted, λ_t^m scatters around 0.5 when ℓ_t^{diff} is a significantly negative value μ_3. However, when the new model improves the accuracy significantly, we prefer a weight smaller than 0.5, and thus the logits z_{new} from the new model is preferred to preserve the accuracy. To achieve this goal, γ^m should be positive to shift the CDF to the right, as illustrated by CDF_2 in Fig. 4. In this way, the new λ_t^m will be smaller than 0.5 on CDF_2 when ℓ_t^{diff} is μ_3.

In summary, the appropriate γ^m should be different for different model update scenarios, which is a trade-off between regression error reduction and accuracy preservation. It will be tuned with validation dataset and analyzed in the hyper-parameter analysis evaluation.

3.3 System-Level Regression Error Reduction

Up to now, we calculate λ_t^m only based on the ASR model losses to reduce regression errors at model level. However, as we have shown in Fig. 2, there are system-level hidden regression errors that are not caused by model-level regression errors. Therefore, we propose to utilize system-level losses (i.e., QA losses) to calculate the ensemble weight λ_t^s. Our approach is intuitive but has challenges to be addressed, because the ASR model outputs the logits and retrieves neighbors token by token, while the whole transcribed question $\hat{q}$ is needed to calculate the QA loss. Therefore, we propose Alg. 3 to estimate system-level loss g_{ij} of token samples in $\mathcal{H}_{train}$ for a given spoken QA system S.

For each training sample in the training dataset $\mathcal{D}_{train}$, Alg. 3 first transcribes it by the ASR model (Line 5), then predicts the answer $\hat{ans}$ for the question with the QA model (Line 6). After that, it calculates the system-level loss $\hat{g}_i$ for every transcribed sample $(\hat{q}_i, p_i)$ with the cross entropy loss of its answer prediction position (Line 7). Finally, it directly approximates the system-level loss for token samples g_{ij} as $\hat{g}_i$, which is its corresponding training sample's system-level loss (Line 11). $\hat{g}_i$ represents the loss of the downstream QA model M_{qa} given the entire question $\hat{q}_i$ transcribed by

M_{asr} from the audio input a_i. Therefore, it is intuitive to use $\hat{g}_i$ as an approximation for the loss of M_{qa} given a specific token in $\hat{q}_i$ predicted by M_{asr}. Notice that this approximation is somewhat coarse, and could be further improved by techniques like token attribution [27].

During the construction of datastore, we execute Alg. 3 with S^{old} and S^{new} to calculate g_{ij}^{old} and g_{ij}^{new}, respectively. During the inference, one intuitive way to obtain λ_t^s is by Eq. 11,

$$\lambda_t^s = \mathcal{E}_s(g_t^{diff}) = \Phi\left(\frac{g_t^{diff} - \mu^s - \gamma^s * \sigma^s}{\sigma^s}\right) \tag{11}$$

where λ_t^s is calculated similar to λ_t^m in Eq. 10 except that it is calculated with system-level loss difference $g_t^{diff} = g_t^{new} - g_t^{old}$. μ^s and σ^s denote the mean and standard deviation of system-level losses difference distribution of all token samples in $\mathcal{H}_{train}$, respectively. γ^s also is a hyper-parameter to shift the CDF of system-level loss difference, which should also be tuned with validation dataset.

However, the inaccuracy of system-level losses approximation may harm the accuracy of the ensemble model significantly. Therefore, we propose an enhanced approach to consider model-level losses and system-level losses together by Eq. 12.

$$\lambda_t^{s+} = \mathcal{E}_{s+}(\ell_t^{diff}) = \Phi\left(\frac{\ell_t^{diff} - \mu^m - (\gamma^m + (1 - 2 * \lambda_t^s) * \rho) * \sigma^m}{\sigma^m}\right) \tag{12}$$

Compared to λ_t^m in Eq. 10, λ_t^{s+} incorporates λ_t^s to shift the CDF of model-level loss difference distribution. For example, if the current token prediction from the new model may cause system-level regression errors, i.e., $\lambda_t^s > 0.5$ calculated by Eq. 11, then $1 - 2 * \lambda_t^s < 0$ holds, and the CDF of ℓ_t^{diff} is shifted to the left. As Fig. 4 shows, shifting CDF_1 to CDF_2 causes λ_t^m increase. Thus, λ_t^{s+} will be larger than λ_t^m, i.e., the ensemble model assigns larger weight on the logits from the old model, then the regression error in this case could be reduced. ρ is the hyper-parameter to balance the influence between model-level losses and system-level losses, because too much weight on system-level losses (especially inaccurate ones) will harm the accuracy of the ensemble ASR model.

4 Evaluation Setups

Research Questions. To evaluate our approach, we design the following six research questions.

- **RQ1 System-Level Regression Error Investigation**: Do system-level regression errors exist and can them be indicated by model-level regression errors?
- **RQ2 Effectiveness Evaluation**: Does RegTrieve reduce more regression errors compared with the baseline approaches without the harm of accuracy?
- **RQ3 Ablation Study**: How does the CDF-based loss ensemble function contributes to RegTrieve?
- **RQ4 Hyper-Parameter Analysis**: What is the impact of hyper-parameters on RegTrieve?
- **RQ5 Efficiency Evaluation**: What is the efficiency of RegTrieve compared with baselines?
- **RQ6 Generalization Evaluation**: Does RegTrieve generalize to various ML systems?

Notice that the first five RQs are conducted on the spoken QA system, while RQ6 is conducted on the multi-sensor fusion perception system.

4.1 Evaluation Setups on the Spoken QA System

Dataset. To evaluate the whole spoken QA system, pure ASR datasets [29] or QA datasets [36] are not enough. Thus, we use HeySQuAD [56], the largest spoken QA dataset up to now, for approach evaluation. It contains 72K/4K human-spoken questions for training and evaluation. Audios in it were created by 12 English speakers from questions in SQuAD1.1 [36] and SQuAD2.0 [35]. We only utilize 48K/1K samples from SQuAD1.1, because lots of questions in SQuAD2.0 are unanswerable,

Table 1. The setups of different update scenarios

ID	Update Scenario	Model	Training Dataset	WER(%)	M.Reg(%)	F1(%)	S.Reg(%)
1	Model Size	$S2T_{small} \rightarrow S2T_{medium}$	$Libri_{all}$	31.69→40.90	7.69	73.85→74.64	7.49
2	Model Size	$S2T_{medium} \rightarrow S2T_{large}$	$Libri_{all}$	31.69→31.44	6.29	73.85→75.90	6.79
3	Model Architecture & Training Data	$S2T_{large} \rightarrow whisper_{tiny}$	$Libri_{all} \rightarrow Whisper^{en}$	41.06→9.87	0.20	72.40→82.40	3.49
4	Prediction Class	$whisper_{tiny}$	$Whisper^{en} \rightarrow Whisper^{mul}$	17.53→13.24	1.80	78.50→80.00	5.59
5	Training Data	$S2T_{small}$	$Libri_{clean} \rightarrow Libri_{all}$	31.57→31.17	4.29	72.42→75.77	5.89
6	Training Data	$S2T_{small}$	$Libri_{all} \rightarrow Libri_{all}$ & $HeySQuAD$	31.17→17.08	0.00	75.77→81.07	1.90
7	Training Step	$S2T_{small}$	$Libri_{all}^{50K} \rightarrow Libri_{all}^{300K}$	44.90→31.17	1.70	72.43→75.77	4.29

Note: M.Reg and S.Reg respectively denote the model-level and system-level regression error rate in percentage. $S2T_{small}$, $S2T_{medium}$ and $S2T_{large}$ are S2T models with different sizes. $Libri_{all}$ and $Libri_{clean}$ denote the LibriSpeech dataset with all data and with only the clean partition, respectively. $Libri_{all}^{50K}$ and $Libri_{all}^{300K}$ denote that the model is trained 50K and 300K steps with the LibriSpeech dataset. $Whisper^{en}$ and $Whisper^{mul}$ are the proprietary dataset used to train the English version and multi-language version of whisper models. $HeySQuAD$ denotes the spoken QA dataset.

which brings difficulties for the system-level loss computation. We randomly select 4K samples from the training dataset as the validation dataset for hyper-parameter tuning.

Models. For the ASR model, we consider two transformer-based model architectures, i.e., *Speech to Text Transformer (S2T)* [52] and *Whisper* [34]. *S2T* was proposed by Meta FAIR, which was pre-trained on LibriSpeech [29]. LibriSpeech is a de-facto standard ASR benchmark that contains 1k hours of English speech from audiobooks. *S2T* is also used by Wu et al. [56] to measure the accuracy of QA models on transcribed audio with noise. *Whisper* was proposed by OpenAI, the state-of-the-art ASR model that supports multi-language transcription. It was pre-trained with proprietary data collected by OpenAI with more than 68K hours audio. 65% of the data represents English transcripts.

For the QA model, we adopt *roberta-large* [25], which is shown to be well-performed with noisy transcribed questions by Wu et al. [56]. We train it with the training data from SQuAD1.1 [36], and keep it unchanged in our model update scenarios.

Update Scenarios. As Table 1 shows, we investigate the following five types of typical model update scenarios with seven specific scenarios which may happen in real-world.

- **Model Size Update.** Update the old model with a new model with more parameters but the same architecture, e.g., the update from $S2T_{small}$ with 29.5M parameters to $S2T_{medium}$ with 71.2M parameters, and $S2T_{medium}$ to $S2T_{large}$ with 1B parameters in Scenario 1 and 2, respectively.
- **Model Architecture Update**. Update the old model with a new model with a different architecture, e.g., the update from $S2T_{large}$ to $whisper_{tiny}$ in Scenario 3.
- **Prediction Class Update.** Update the old model with a new model supporting more prediction classes, e.g., the update from an English-only model to a multi-language model in Scenario 4. These two models share the same architecture and size, but are trained with different training data.
- **Training Data Update.** Update the old model with a new model trained with more data. For example, the old model is pre-trained only with clean partition of LibriSpeech, but the new model is pre-trained with all of LibriSpeech in Scenario 5. Another scenario is to perform further fine-tuning with task specific training data, e.g., the $S2T_{small}$ is further fine-tuned with training data from HeySQuAD in Scenario 6.
- **Training Step Update.** Update the old model with a new model trained with the same training data but more training steps. For example, the old model is pre-trained with 50K steps while the new model is pre-trained with 300K steps with LibriSpeech in Scenario 7.

We directly use the released model checkpoints from Hugging Face [42] in the first four scenarios, while train the model in the last three scenarios. Table 1 indicates that the magnitude of model-level regression error rate does not fully reflect system-level regression error rate. For example, although

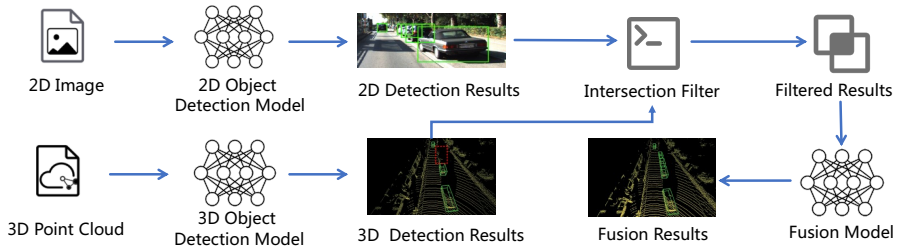


Fig. 5. An illustration of the multi-sensor fusion perception system

Scenario 4 has a 2.49% lower model-level regression error rate than Scenario 5, its system-level regression error rate is lower by only 0.3%. It is consistent with Eq. 8, and will be analyzed in Sec. 5.1. Further, system-level regression error rate is larger that model-level one across almost all scenarios.

Baseline Approaches. There are two mainstream types of approaches to reduce regression errors, training-based and ensemble-based approaches [23]. Training-based approaches are time-consuming and have been shown by Li et al. [23] to significantly reduce model accuracy. Therefore, we selected ensemble-based approaches as baselines. We did not select Bayesian-based ensemble because, when the number of predicted classes is large (in our token prediction scenario), Bayesian estimation of ensemble weights becomes inaccurate due to the large prior and posterior matrix, greatly affecting model accuracy, as already revealed by Li et al. [23]. Besides, NeuRecover [48] is not open-source due to company policies. In summary, we select the following baselines.

- **Simple Approaches.** We select simple approaches with heuristics, including *simple average (Avg)* and *taking max belief (Max)* of logits between z_{old} and z_{new} predicted by two ASR models.
- **Uncertainty-Based Approaches.** We select two uncertainty-based approaches proposed by Li et al. [23]. Specifically, for a given testing sample, they estimate the uncertainty of a model by the variance from multiple forward passes via *Dropout* or random noise (*Pertub*). Then, they compute the ensemble weight λ_t based on the uncertainty of the old and new model. We adopt their default hyper-parameters for comparison. It has been shown that they are the state-of-the-art to reduce model-level regression errors in image classification tasks [23].

Implementation. The datastore $\mathcal{H}_{train}$ contains millions of token-level samples, thus we use FAISS [15] to index and search neighbors efficiently. The last decoder hidden states of samples are used as the keys in the datastore, which are supposed to capture the semantic meaning of current sample effectively [17, 20]. The number of retrieved neighbors K is set to 20. We set the threshold τ_m and τ_s in Eq. 3 and 5 to 0.3. γ^m in Eq. 10 and γ^s in Eq. 11 are tuned with the HeySQuAD validation dataset. Once the most appropriate γ^m and γ^s for Pareto Improvement are decided, ρ in Eq. 12 is tuned with fixed γ^m and γ^s. The experiments were conducted on a server with Intel (R) Core (TM) i9-10980XE CPU @ 3.00GHz, 128GB RAM, and 4 NVIDIA GeForce RTX 3090 GPUs.

The ensemble of models with different vocabulary corpus requires vocabulary alignment, such as Scenario 3 and 4, because logits z_{old} and z_{new} should be summed in a weighted way directly. We adopt the vocabulary alignment from existing work [51], which utilizes *edit distance* to match vocabularies between different models. The vocabulary of the old model is mapped into the vocabulary of the new model, which causes WER of the old model increase. For example, WER of $S2T_{large}$ increases from 31.44% (new model in Scenario 2) to 41.06% (old model in Scenario 3) with token alignment.

4.2 Evaluation Setups on the Multi-Sensor Fusion Perception System

To answer RQ6, we conduct experiments on CLOCs (Camera-LiDAR Object Candidates Fusion) [30], a representative multi-sensor fusion perception system. As shown in Fig. 5, it consists of three ML components (i.e., a 2D object detection model, a 3D object detection model, and a fusion model) and one

non-ML component (i.e., the intersection filter). First, a 2D image from a camera and a 3D point cloud data from a LiDAR are processed by the respective 2D and 3D object detection models to generate 2D and 3D detected bounding boxes. Then, the intersection filter applies rule-based filtering to identify overlapping bounding boxes. Finally, a CNN-based fusion model, trained on labeled data, merges the filtered bounding boxes to produce the final detection result.

Metrics. In object detection tasks, true positives and false positives are calculated based on the IOU (Intersection over Union) between predicted and ground truth bounding boxes. Recall, precision, and regression errors are thus all computed based on IOU. We use recall to measure the accuracy of the 2D object detection model, as any redundant boxes will be filtered out by the intersection filter, making precision less important. F1 score is used to measure the system accuracy.

Dataset. We adopt the widely used 3D object detection benchmark KITTI [16]. It includes LiDAR point clouds and camera images, and consists of 7,481 training samples and 7,518 testing samples.

Models. We use Cascade R-CNN [11] and YOLO-v5 [47] as the 2D object detection models, SECOND [59] as the 3D object detection model, and CLOCs fusion model [30] as the fusion model.

Update Scenarios. Only the 2D object detection model was updated, while other components remained unchanged. Two update scenarios were considered: (1) model architecture update, i.e., replacing R-CNN with YOLOv5 (Scenario 8); and (2) training step update, i.e., replacing R-CNN trained for 1 epoch with R-CNN trained for 12 epochs (Scenario 9).

Baseline Approaches. Let $B^{old} = \{b_1^{old}, b_2^{old}, \ldots, b_m^{old}\}$ be the bounding boxes detected by the old model on a sample, $B^{new} = \{b_1^{new}, b_2^{new}, \ldots, b_n^{new}\}$ be the bounding boxes detected by the new model, and $IOU(b_i^{old}, b_j^{new})$ be the IOU between b_i^{old} and b_j^{new}. Here, the goal of regression error reduction is to compute the refined bounding box set $B^{'}$ according to B^{old} and B^{new}, such that $Reg(B^{old}, B^{'}) < Reg(B^{old}, B^{new})$. The bounding boxes in B^{old} and B^{new} with an IOU greater than 0.8 are denoted as $B^{old}_{overlapped}$ and $B^{new}_{overlapped}$ (whose size is denoted as o), respectively. The remaining parts are denoted as B^{old}_{left} and B^{new}_{left}. We use Eq. 13 to merge $B^{old}_{overlapped}$ and $B^{new}_{overlapped}$.

$$B_{selected} = \bigcup_{i=1}^{o} \begin{cases} B^{old}_{overlapped,i} & \text{with the probability of } \lambda \\ B^{new}_{overlapped,i} & \text{with the probability of } 1-\lambda \end{cases} \tag{13}$$

That is to say, each overlapped bounding box in $B^{'}$ is selected from B^{old} and B^{new} based on the probability λ. λ plays a role similar to the logits ensemble weight in spoken QA scenarios. Based on $B_{selected}$, B^{old}_{left}, and B^{new}_{left}, we construct the following baselines for obtaining the final B'.

- **Simple Approaches.** We select two simple approaches (respectively named *Old* and *New*), where λ is set to 0.5, and the remaining bounding boxes from the old model B^{old}_{left} and the new model B^{new}_{left} are respectively unioned with $B_{selected}$; i.e., $B^{'} = B_{selected} \cup B^{old}_{left}$, and $B^{'} = B_{selected} \cup B^{new}_{left}$.
- **Uncertainty-Based Approaches.** We select one uncertainty-based approach (named *Perturb*) by adding random noise to the input image data, calculating λ based on the variance of the predicted bounding boxes, and unioning B^{old}_{left} and $B_{selected}$. Since R-CNN and YOLO-v5 do not have explicit dropout layers, we cannot use the dropout-based approach as a baseline.

RegTrieve Adaption. Similar to *Perturb*, RegTrieve also needs to estimate λ, but it is based on the loss of retrieved neighbors. We use the commonly used localization loss in object detection tasks to compute the model-level loss, and then calculate the λ^m based on the CDF of loss difference, just as in the spoken QA scenarios. The system-level loss is obtained by passing the 2D model's object detection results through the entire pipeline, calculating the fusion model's loss. Finally, the λ^s is computed based on the system-level loss.

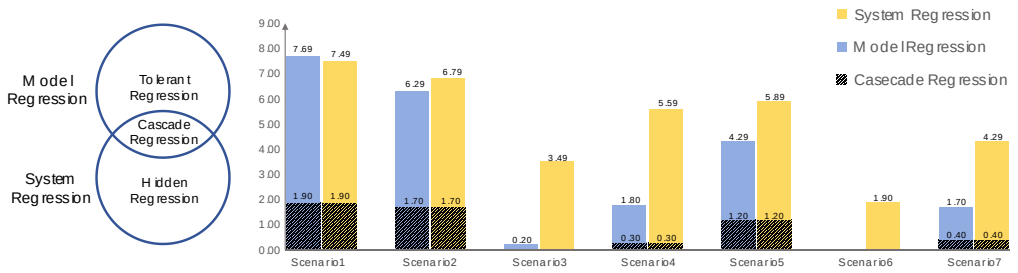


Fig. 6. The distribution of model-level regression errors and system-level regression errors

5 Evaluation Results

5.1 RQ1: System-Level Regression Errors Investigation

We first investigate the relation between model-level regression errors and system-level regression errors in our seven update scenarios. The right part of Fig. 6 shows model-level, system-level, and cascade (shaded areas) regression error rates across different update scenarios. In Scenario 1, 2 and 5, cascade regression errors account for less than 20% of the system-level regression errors, while in other scenarios, they are under 10%. This indicates that only a small portion of system-level regression errors are directly caused by model-level regression errors, with the rest is attributed to hidden regression errors. Similarly, only lower than 20% of model-level regression errors result in system-level regression errors, others are tolerant regression errors.

Summary. Only a small portion of system-level regression errors are caused by model-level regression errors, and vice versa. Therefore, for ML systems involving multiple ML models, the reduction of system-level regression errors require separate investigation.

5.2 RQ2: Effectiveness Evaluation

We compare RegTrieve against baseline approaches across seven update scenarios. Table 2 reports the comparison results, where the *Orig* column lists the original results after model update (i.e., before applying reduction techniques), the *Model*, *System* and *System+* columns under RegTrieve represent the results of three settings of RegTrieve, calculated by Eq. 10, 11 and 12, respectively.

Pareto Improvement. The goal of regression error reduction approaches is not to maximize the F1 score but to reduce system-level regression errors without compromising the F1 score, achieving a *Pareto improvement*. Overall, RegTrieve achieves *Pareto Improvement* in all scenarios except for a slightly F1 score degradation in Scenario 3, whereas *Perturb* and *Avg* only achieve it in two scenarios, and *Max* and *Dropout* only succeed in one. In scenarios where the new system shows significant F1 score improvement over the old system, such as Scenario 3, 5 and 6, reducing regression errors while maintaining accuracy is more challenging due to the higher risk of F1 score degradation from introducing the old model's logits. In such scenarios, our approach estimates higher losses for the old model and reduces the weight assigned to the old model's logits, thus preserving F1 score. Compared to the image classification task evaluated by Li et al. [23], the ASR task is more complex, as the number of predicted tokens is significantly larger than the number of image categories, making *Dropout* and *Perturb* estimate the uncertainty less accurately.

Regression Error Reduction. Table 2 shows that in almost all scenarios, RegTrieve reduces more regression errors compared to baseline approaches, unless baseline approaches sacrifice F1 score. For example, in Scenario 1 and 2, *System+* reduces system-level regression error rate from 7.49% and 6.79% to 1.50% and 3.09%, respectively, whereas the baseline approaches only reduce it to a minimum of 4.69% (*Max*) and 4.19% (*Avg*). In Scenario 7, *Avg* reduces system-level regression error rate to 2.59%, which is lower than the 2.99% achieved by *Model*, but at the cost of reducing F1

Table 2. The results of different approaches in the spoken QA system

Scenario	Metric(%)	Orig	Simple		Uncertainty		RegTrieve			Ablation	
			Max	Avg	Dropout	Pertub	Model	System	System+	M.Loss	S.Loss
1	WER	40.90	31.65	31.10	32.25	31.69	27.53	26.98	**28.32**	27.60	27.65
	M. Reg	7.68	2.20	2.20	3.19	2.50	0.90	0.70	**0.60**	1.60	1.10
	F1	74.64	74.47	75.21	73.82	75.26	74.71	74.72	**74.70**	75.47	75.26
	S. Reg	7.49	4.69	4.99	5.49	4.79	2.99	3.79	**1.50**	4.69	4.89
2	WER	31.44	26.70	25.43	25.68	26.09	**24.47**	24.22	**24.26**	24.17	24.20
	M. Reg	6.29	2.10	1.30	2.30	1.60	**1.10**	1.30	**1.10**	1.20	1.50
	F1	75.90	75.78	76.12	74.83	76.10	75.96	76.32	**76.12**	75.94	75.98
	S. Reg	6.79	4.69	4.19	5.89	4.99	3.09	4.19	**3.09**	4.79	4.39
3	WER	9.87	31.47	31.31	18.19	27.05	11.01	11.84	11.09	13.24	30.06
	M. Reg	0.20	1.30	1.50	0.90	0.60	0.50	0.70	0.50	0.70	2.10
	F1	82.44	77.66	75.67	81.42	78.40	82.07	82.09	82.13	81.03	75.28
	S. Reg	3.49	2.99	5.39	3.89	3.29	3.79	3.39	3.79	3.99	7.19
4	WER	13.24	**12.23**	12.81	**12.75**	12.96	17.06	15.58	15.09	12.30	12.68
	M. Reg	1.80	**0.40**	0.70	**0.40**	0.70	0.40	0.40	0.80	1.10	0.60
	F1	80.00	81.42	81.55	81.62	81.08	80.06	**81.00**	81.13	81.46	81.68
	S. Reg	5.59	2.10	2.50	1.90	2.89	1.90	**1.70**	2.10	3.09	2.79
5	WER	31.17	28.63	28.69	27.32	28.47	**26.84**	26.88	**27.15**	26.43	27.34
	M. Reg	4.29	2.40	2.30	1.80	2.20	**1.20**	1.60	**1.20**	1.30	1.40
	F1	75.77	75.80	75.59	73.91	75.65	75.85	75.92	**75.85**	76.25	75.62
	S. Reg	5.89	4.39	4.49	6.39	4.59	3.89	4.69	**3.79**	4.79	4.69
6	WER	17.08	26.43	24.51	20.10	21.20	16.85	17.95	16.85	18.13	18.86
	M. Reg	0.00	0.90	0.20	0.30	0.20	0.10	0.10	0.10	0.20	0.10
	F1	81.07	78.49	78.87	79.35	79.67	**81.85**	**81.43**	**81.85**	81.29	80.58
	S. Reg	1.90	1.40	1.20	1.70	1.70	**1.10**	**1.10**	**1.10**	1.40	1.60
7	WER	31.17	35.62	33.90	33.02	35.65	31.48	**31.04**	31.41	31.53	31.13
	M. Reg	1.70	1.10	1.10	0.60	1.30	1.00	**0.80**	1.20	1.10	0.90
	F1	75.77	74.76	75.51	74.95	74.63	75.94	**75.99**	76.02	75.74	75.99
	S. Reg	4.29	2.99	2.59	3.09	2.69	3.19	**2.99**	3.29	3.39	3.09

Note: The bold values indicate the approaches that achieve the greatest reduction in model-level or system-level regression errors without compromising the WER or F1 score in each scenario.

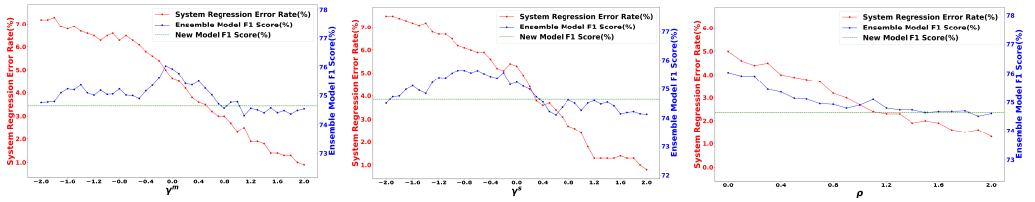
score from 75.77% to 75.50%. In contrast, *Model* maintains an F1 score of 75.99%. Among the three settings of RegTrieve, *System+* generally achieves the best performance because it combines both model-level and system-level losses. However, in Scenario 3 and 4, the old ASR model requires token alignment, which makes the estimation of model-level losses less accurate. As a result, *System*, which relies solely on system-level losses, can reduce more regression errors in these scenarios.

Summary. RegTrieve consistently outperforms baseline approaches in reducing regression errors while maintaining system accuracy. Numerically, RegTrieve reduces more regression errors than all baselines by 20.43% in average. RegTrieve achieves Pareto Improvement in all but one scenario, with *System+* and *System* demonstrating the best performance in different scenarios.

5.3 RQ3: Ablation Study

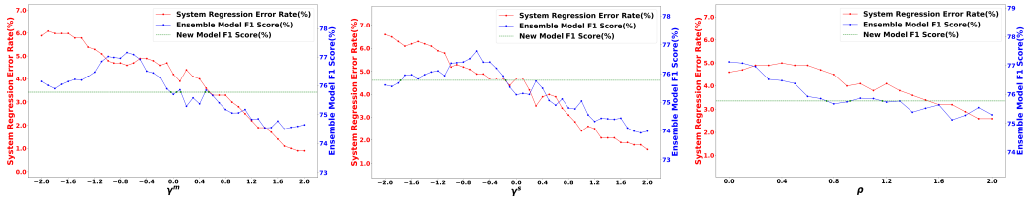
RegTrieve primarily consists of two parts, i.e., estimating the testing sample's loss using the losses from similar training samples, and computing the ensemble weight based on CDF of the loss difference. We investigate the contribution of the CDF-based ensemble function by simply using the ratio of old loss to new loss as the ensemble weight. Specifically, the ensemble weight is calculated by $\lambda_m^{'} = \ell_t^{new}/(\ell_t^{new} + \ell_t^{old})$ and $\lambda_s^{'} = g_t^{new}/(g_t^{new} + g_t^{old})$, respectively denoted as *M. Loss* and *S. Loss* in the last two columns in Table 2. *M. Loss* and *S. Loss* achieve Pareto Improvement in five scenarios (i.e., Scenario 1, 2, 4, 5 and 6) and four scenarios (i.e., Scenario 1, 2, 4 and 7), respectively, exceeding the number of scenarios where the baseline approaches achieve. However, the average reduction in regression errors for *M. Loss* and *S. Loss* is significantly less than that of RegTrieve.

Summary. RegTrieve without the CDF-based ensemble function (*M. Loss* and *S. Loss*) still achieves Pareto Improvement in more scenarios than all baseline approaches. However, the CDF-based ensemble function is important for more reduction of system-level regression errors.



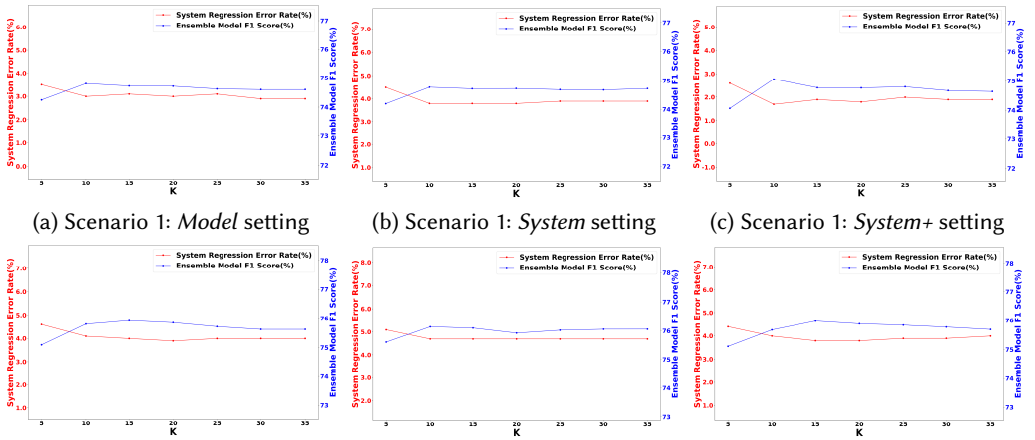
(a) Analysis of γ^m (b) Analysis of γ^s (c) Analysis of ρ

Fig. 7. The impact of model ensemble hyper-parameters in Scenario 1



(a) Analysis of γ^m (b) Analysis of γ^s (c) Analysis of ρ

Fig. 8. The impact of model ensemble hyper-parameters in Scenario 5



(a) Scenario 1: *Model* setting (b) Scenario 1: *System* setting (c) Scenario 1: *System+* setting

(d) Scenario 5: *Model* setting (e) Scenario 5: *System* setting (f) Scenario 5: *System+* setting

Fig. 9. The impact of K in Scenario 1 and Scenario 5

5.4 RQ4: Hyper-Parameter Analysis

We analyze the impact of hyper-parameters on the performance of RegTrieve. There are three model ensemble hyper-parameters to configure for the *Model*, *System* and *System+* settings, namely γ^m, γ^s and ρ, respectively. Next, we measure the impact of K (i.e., the number of neighbors retrieved to estimate the ensemble weight). Due to space limitation, we only demonstrate the impact of these hyper-parameters for Scenario 1 and Scenario 5. These two scenarios represent cases where F1 score improvement of the new system over the old system is small (Scenario 1) and large (Scenario 5), respectively. Results for other scenarios are also available at our website [7].

Fig. 7 and 8 show the impact of model ensemble hyper-parameters in Scenario 1 and Scenario 5, respectively. As for the impact of γ^m on the *Model* setting, as it increases, system-level regression error rate continues to decrease in both scenarios. This is because a larger γ^m assigns more weight to the old model during model ensemble, thus reducing regression errors. F1 score of the ensemble model in both scenarios initially increases and then decreases, because the ensemble model could effectively combine the logits of the new and old models when γ^m is around 0, achieving a higher

Table 3. The initialization overhead of RegTrieve

Scenario	Datastore Storage		I/O Time (in seconds)		
	Model	System	Model	Dataset	Datastore
1	4.4G	12.4M	0.281	3.841	1.927
2	7.2G	12.4M	0.591	3.817	2.922
3	7.5G	12.1M	2.363	6.268	2.599
4	3.7G	11.8M	1.724	6.448	1.733
5	2.6G	11.8M	0.241	3.823	1.200
6	2.6G	11.8M	0.244	3.873	1.185
7	2.6G	11.8M	0.254	3.853	1.200

Note: *Model* and *System* under the *Datastore Storage* column represent the sizes of the datastores when using model-level and system-level loss in RegTrieve, respectively. *I/O Time* indicates the time to load *Model*, *Dataset* and *Datastore*.

Table 4. The inference time of different approaches for a testing sample (in seconds)

Scenario	Old M.	New M.	Simple	Uncertainty		RegTrieve
			Max/Avg	Dropout	Pertub	All Settings
1	0.016	0.019	0.106	1.307	1.281	0.464
2	0.017	0.020	0.197	2.136	2.213	0.652
3	0.017	0.022	0.295	2.746	2.957	0.348
4	0.013	0.022	0.238	2.214	2.344	0.264
5	0.016	0.016	0.081	0.939	0.980	0.290
6	0.017	0.014	0.086	0.931	0.945	0.247
7	0.019	0.017	0.098	1.064	1.039	0.255

Note: *Old M.* and *New M.* denotes the inference time of the old ASR model and new ASR model. The inference time for *Max* and *Avg* is very close, and thus we combine them into one column. The same also applies to the three settings of RegTrieve.

F1 score. When γ^m becomes larger, F1 score in Scenario 5 decreases more rapidly than that in Scenario 1 due to the larger performance gap between the old and new models. The impact of γ^s on the *System* setting follows a similar trend to the impact of γ^m.

When evaluating the impact of ρ on the *System+* setting, we select γ^m (which maximizes F1 score) and γ^s (which minimizes regression error rate) based on previous results. As shown in Fig. 7 and 8, both F1 score and system-level regression error rate decrease as ρ increases. This is because a larger ρ increases the influence of *System*, which has lower F1 score and system-level regression error rate, on *System+*. Finally, by performing hyper-parameter tuning on the validation set, we select γ^m, γ^s and ρ values that do not decrease F1 score of the new system while minimizing the regression error rate, and report the corresponding results in Table 2.

As shown in Fig 9, under the three settings of RegTrieve, the F1 score initially increases with K and then stabilizes. Similarly, the system regression error rate decreases with increasing K and then stabilizes. This trend is expected, as a larger K means more neighbors, leading to more accurate estimates of model accuracy and ensemble weights. However, beyond 15 or 20 neighbors, the newly added neighbors are increasingly distant from the current sample and contribute less to the final result. Therefore, setting K to 20 in previous experiments is a reasonable choice.

Summary. Increasing γ^m and γ^s consistently reduces system-level regression errors, while increasing first and then reducing F1 score. Increasing ρ in *System+* reduces both F1 score and system-level regression errors due to the growing influence of *System*. Increasing K to 15 or 20 improves system accuracy and reduces system-level regression errors.

5.5 RQ5: Efficiency Evaluation

We first measure the initialization overhead, including the data repository size and the I/O time for loading the data repository, and then measure the inference time of RegTrieve.

Table 5. The results of different approaches in the multi-sensor fusion perception system

Scenario	Metric(%)	Orig	Simple		Uncertainty	RegTrieve		
			New	Old	Pertub	Model	System	System+
8	M. Recall	95.26	95.23	87.37	95.25	96.86	96.88	**96.89**
	M. Reg	4.15	4.08	0.12	4.11	1.20	1.23	**1.17**
	S. F1	84.60	82.93	82.7	84.60	84.36	84.41	84.40
	S. Reg	2.64	2.02	0.80	2.23	1.76	1.70	1.72
9	M. Recall	87.13	86.88	74.88	87.09	**87.95**	88.02	88.03
	M. Reg	1.92	1.90	0.15	1.90	**0.95**	1.01	1.01
	S. F1	83.01	83.13	81.93	82.99	83.26	**83.02**	**83.03**
	S. Reg	1.01	0.86	0.14	0.92	0.80	**0.78**	**0.78**

Note: The bold values indicate the approaches that achieve the greatest reduction in model-level or system-level regression errors without compromising the recall or F1 score in each scenario.

Initialization Overhead. As shown in Table 3, the datastore size of RegTrieve ranges from 2GB to 8GB in different scenarios, which can typically be fully loaded into the memory of mainstream machines. Additionally, it can be observed that the time taken to load the datastore is similar to that of loading the model and dataset. The latter two are essential for normal model inference, and thus the additional I/O cost introduced by RegTrieve is relatively small. Moreover, the I/O time during initialization is minimal compared to the total inference time (from 200 to 700 seconds).

Inference Time. We report only the inference time of the ASR model with different approaches in Table 4, while the QA model's inference time is not included since it remains unchanged in our experiments. For a single testing sample, RegTrieve's average inference time is 0.36 seconds, which is only about 20% of the time required by *Dropout* (1.62 seconds) and *Perturb* (1.69 seconds). Compared to *Max/Avg*, RegTrieve introduces only about twice the time overhead due to retrieval and loss ensemble, whereas uncertainty-based approaches require multiple forward passes (10 in this case) for uncertainty estimation. The inference time in Scenario 3 and 4 is larger than that in other scenarios due to the additional time required for vocabulary mapping. Note that the time required by *Max/Avg* is significantly greater than that of the old system and the new system, as we implement a custom greedy decoding logic for easier logits ensemble. Replacing this with the built-in decoding function from Transformers [44] library will further speed up all reduction approaches.

Summary. The datastores of RegTrieve range from 2GB to 8GB, which could be fully loaded into memory for later inference. Therefore, RegTrieve only needs about 20% of the time required by uncertainty-based approaches for inference, introducing twice the overhead of *Max/Avg* approaches.

5.6 RQ6: Generalization Evaluation

We report the adaption effort and generalization effectiveness when applying RegTrieve to another ML system that contains non-ML components, i.e., the multi-sensor fusion perception system.

Adaption Effort. During adaptation, the overall framework of RegTrieve is the same to that of the spoken QA system, as both rely on loss-based model accuracy estimation to compute ensemble weights. However, we need to adapt the methods for calculating the loss and merging bounding boxes that differ from the spoken QA system. All baselines also need to be adapted for bounding box merging. Notice that the *Dropout* approach cannot be adapted because the models lacks a dropout layer. Overall, RegTrieve requires similar small adaptation efforts, compared with baselines.

Generalization Effectiveness. As shown in Table 5, all three settings of RegTrieve reduce more regression errors without compromising model-level recall, compared to baselines. Among them, *System+* reduces the most regression errors both in Scenario 8 and Scenario 9. Since *Old* retains many non-overlapped bounding boxes from the old model, it has fewer regression errors. However, it does not consider enough results from the new model, leading to a much lower accuracy. Although in Scenario 8, RegTrieve causes a slight decrease in system-level F1, the reduction is minimal compared to baselines, while also reducing a relatively larger number of system-level regression errors.

Summary. Compared with baselines, RegTrieve requires similar efforts for adaption, and shows promising generalization capability by reducing 14.06% more system-level regression errors in the multi-sensor fusion perception system on average.

5.7 Threats

First, RegTrieve is evaluated on a spoken QA system and a multi-sensor fusion perception system, and thus its effectiveness on other complex ML systems, remains uncertain. We have only considered scenarios involving a single upstream model update. More complex cases, such as system regression errors arising from multiple models being updated simultaneously or updates involving downstream models, require further evaluation and research. Second, the inadequacy of accuracy evaluation metrics (e.g., WER and F1 score) may lead to bias in computing regression errors during experiments. Future work should investigate whether improving metrics influences our findings, particularly the observation that model-level and system-level regression errors do not always occur simultaneously. Third, we have only evaluated the ensemble ASR models using greedy decoding. The performance on other decoding strategies, such as *Multinomial Sampling* [43] or *Beam-Search Decoding* [14], remains unknown. However, the evaluation results from greedy decoding reflect the quality of the ensemble model's predicted logits and could serve as a good indicator for the performance across other decoding strategies. Finally, in our evaluation scenarios, the datastore can be fully loaded into memory, and thus retrieving samples introduces minimal overhead. If the datastore becomes too large to fit into memory, retrieval operations would incur additional disk I/O overhead. In such cases, RegTrieve could easily benefit from techniques designed to optimize RAG, such as *Caching Embeddings* [45] and *Contextual Compression* [46].

6 Related Work

ML Regression Errors. Regression bugs have been widely investigated by the SE community [60], including regression testing [13, 60], debugging [53] and automatic regression bug dataset construction [37]. Recently, the ML community has started to notice regression errors during model updates, due to potential negative influences on human trust [8] in human-AI collaboration scenarios and downstream modules in ML systems [38]. Some studies refer this to backward compatibility problem [8, 38, 50], but it is essentially the same as ML regression errors [23, 58]. DRFuzz [62] generates inputs that trigger diverse regression errors. Techniques to reduce regression errors could be divided into two categories, training-based approaches and ensemble-based approaches.

Training based approaches aim to add additional loss term during training of the updated model to align its output with old model's output without sacrificing its accuracy too much. For example, Yan et al. [58] proposed focal distillation to reduce the KL divergence of regression error samples between the updated and old model in image classification tasks. Xie et al. [57] and Caciolai et al. [10] extended focal distillation to natural language classification tasks and spoken language labeling with revised distillation loss, respectively. We did not adopt this approach for two reasons. First, training the updated model is costly and may not be feasible in practice (e.g., when the updated model is provided by cloud APIs). Second, Zhu et al. [3] demonstrated that training-based approaches essentially average the old model and updated model. Furthermore, Li et al. [23] showed that training-based approaches significantly harm the accuracy of the updated model.

Ensemble-based approaches combine the prediction results of the updated model and the old model to reduce regression errors. For example, Trauble et al. [50] proposed a Bayesian-based ensemble approach to iteratively update the prediction posterior. NeuRecover [48] adjusted the weights of the model that can safely correct the regressions by comparing the training history between old models and new models. Li et al. [23] aligned the uncertainty of predictions of old model and updated model via uncertainty estimation before the model ensemble. Their experiment results

show that ensemble-based approaches could achieve better results than training-based approaches. Therefore, our retrieval-enhanced approach builds upon the ensemble approach.

To the best of our knowledge, there has been no comprehensive evaluation or reduction of regression errors at system level. Srivastava et al. [38] highlighted the potential impact of ML regression errors in ML system, and conducted a conceptual analysis of an OCR system. However, they neither evaluated the regression errors nor proposed any reduction approaches for a real-world ML system.

Study of ML Systems. While much of the attention has been on ML models [32, 64], system-level analysis has received less attention [18]. We begin by discussing the study of general challenges that arise from integrating multiple models in complex ML systems. Peng et al. [33] investigated the integration of ML models in Apollo by analyzing how these models interact with the system. Cao et al. [12] explored the complexity and testing challenges due to module interactions in Rasa. Moreover, Abdessalem et al. [1, 2] studied the feature interaction failures in self-driving systems. Apel et al. [6] also discussed feature interactions in ML systems. In addition, several attempts have been proposed to address the issue of ML component entanglement [5], including applying metamorphic testing on systems with two ML components [63], troubleshooting failures in systems with three ML components using human intellect [28], and decomposing errors in systems with two or three ML components [55]. Similar to them, our work also takes the perspective of a whole system.

Next, we summarize the existing efforts aimed at optimizing the overall performance of ML systems. Amershi et al. [5] and Bernardi et al. [9] reported that models can be complexly entangled to cause non-monotonic errors [5], and model quality improvement does not necessarily indicate system value gain [9]. Takanobu et al. [40] showed that the component-wise evaluation results are not always consistent with the overall performance of task-oriented dialogue systems. To address this, Lin et al. [24] proposed a joint system-wise optimization method to improve system performance in dialogue systems. Similarly, Li et al. [22] and Wu et al. [56] found that ASR errors severely affect the accuracy of downstream QA models. Su et al. [39] suggested using contextualized word representations in QA models to mitigate the detrimental effects of ASR errors. You et al. [61] employed knowledge distillation to align the ASR model with the expectations of QA models.

In summary, it is important to reduce regression errors from a system perspective, and our work complements previous research, as they do not consider regression errors at system level.

Retrieval-Based Approaches. Retrieval-based augmentation techniques have been used to enhance language model without fine-tuning on domain specific data [17, 20, 41]. They can be categorized into two types, i.e., retrieval enhancement for inputs [26, 31] and outputs [17, 20, 41]. Our work is inspired by the latter. Rather than directly augmenting the retrieved logits from training data, we use the retrieved neighbors to calculate weights for ensembling the old and updated models.

7 Conclusions and Data Availability

We formulate the problem of system-level regression errors, and propose a retrieval-enhanced ensemble approach, RegTrieve, to reduce both model-level and system-level regression errors. Experiments in spoken QA system demonstrate that RegTrieve reduces system-level regression errors by 20.43% more than all baseline approaches, with little or even no loss in system F1 score. All the data and code of RegTrieve have been released at our website [7] to foster future research.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant No. 62332005 and 62372114). This research was also supported by the Ministry of Education, Singapore under its Academic Research Fund Tier 2 (Proposal ID: T2EP20223-0043; Project ID: MOE-000613-00). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

References

[1] Raja Ben Abdessalem, Annibale Panichella, Shiva Nejati, Lionel C. Briand, and Thomas Stifter. 2018. Testing Autonomous Cars for Feature Interaction Failures Using Many-Objective Search. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 143–154.

[2] Raja Ben Abdessalem, Annibale Panichella, Shiva Nejati, Lionel C. Briand, and Thomas Stifter. 2020. Automated Repair of Feature Interaction Failures in Automated Driving Systems. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 88–100.

[3] Zeyuan Allen-Zhu and Yuanzhi Li. 2023. Towards Understanding Ensemble, Knowledge Distillation and Self-Distillation in Deep Learning. In *Proceedings of the 11th International Conference on Learning Representations*.

[4] Samson Alva and Vikram Manjunath. 2019. Strategy-proof Pareto-improvement. *Journal of Economic Theory* 181 (2019), 121–142.

[5] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software Engineering for Machine Learning: A Case Study. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice*. 291–300.

[6] Sven Apel, Christian Kästner, and Eunsuk Kang. 2022. Feature Interactions on Steroids: On the Composition of ML Models. *IEEE Software* 39, 3 (2022), 120–124.

[7] Anonymous Authors. 2024. *RegTrieve: Reducing System-Level Regression Errors for Machine Learning Systems.* Retrieved September 12, 2024 from https://sites.google.com/view/regtrieve/home

[8] Gagan Bansal, Besmira Nushi, Ece Kamar, Daniel S Weld, Walter S Lasecki, and Eric Horvitz. 2019. Updates in human-ai teams: Understanding and addressing the performance/compatibility tradeoff. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 2429–2437.

[9] Lucas Bernardi, Themistoklis Mavridis, and Pablo Estevez. 2019. 150 Successful Machine Learning Models: 6 Lessons Learned at Booking.Com. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1743–1751.

[10] Andrea Caciolai, Verena Weber, Tobias Falke, Alessandro Pedrani, and Davide Bernardi. 2023. Regression-free model updates for spoken language understanding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*. 538–551.

[11] Zhaowei Cai and Nuno Vasconcelos. 2019. Cascade R-CNN: High quality object detection and instance segmentation. *IEEE transactions on pattern analysis and machine intelligence* 43, 5 (2019), 1483–1498.

[12] Junming Cao, Bihuan Chen, Longjie Hu, Jie Gao, Kaifeng Huang, Xuezhi Song, and Xin Peng. 2023. Characterizing the Complexity and Its Impact on Testing in ML-Enabled Systems : A Case Sutdy on Rasa. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution*. 258–270.

[13] Lingchao Chen, Foyzul Hassan, Xiaoyin Wang, and Lingming Zhang. 2020. Taming behavioral backward incompatibilities via cross-project testing and analysis. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 112–124.

[14] Ronan Collobert, Awni Hannun, and Gabriel Synnaeve. 2019. A fully differentiable beam search decoder. In *Proceedings of the International Conference on Machine Learning*. 1341–1350.

[15] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. *arXiv preprint arXiv:2401.08281* (2024).

[16] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3354–3361.

[17] Qi Guo, Xiaohong Li, Xiaofei Xie, Shangqing Liu, Ze Tang, Ruitao Feng, Junjie Wang, Jidong Ge, and Lei Bu. 2024. FT2Ra: A Fine-Tuning-Inspired Approach to Retrieval-Augmented Code Completion. *arXiv preprint arXiv:2404.01554* (2024).

[18] Christian Kästner. 2022. *Machine Learning in Production: From Models to Systems.* Retrieved September 8, 2024 from https://ckaestne.medium.com/machine-learning-in-production-from-models-to-systems-e1422ec7cd65

[19] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. 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*, Vol. 1. 2.

[20] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through Memorization: Nearest Neighbor Language Models. In *Proceedings of the 8th International Conference on Learning Representations*.

[21] Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. 2019. SUMBT: Slot-utterance matching for universal and scalable belief tracking. *arXiv preprint arXiv:1907.07421* (2019).

[22] Chia-Hsuan Li, Szu-Lin Wu, Chi-Liang Liu, and Hung-yi Lee. 2018. Spoken SQuAD: A Study of Mitigating the Impact of Speech Recognition Errors on Listening Comprehension. In *Proceedings of the 19th Annual Conference of the International Speech Communication Association*. 3459–3463.

[23] Zenan Li, Maorun Zhang, Jingwei Xu, Yuan Yao, Chun Cao, Taolue Chen, Xiaoxing Ma, and Jian Lü. 2023. Lightweight Approaches to DNN Regression Error Reduction: An Uncertainty Alignment Perspective. In *Proceedings of the 45th IEEE/ACM International Conference on Software Engineering*. 1187–1199.
[24] Zichuan Lin, Jing Huang, Bowen Zhou, Xiaodong He, and Tengyu Ma. 2021. Joint System-Wise Optimization for Pipeline Goal-Oriented Dialog System. *arXiv preprint arXiv:2106.04835* (2021).
[25] Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).
[26] Shuai Lu, Nan Duan, Hojae Han, Daya Guo, Seung-won Hwang, and Alexey Svyatkovskiy. 2022. ReACC: A Retrieval-Augmented Code Completion Framework. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 6227–6240.
[27] Ali Modarressi, Mohsen Fayyaz, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. 2022. GlobEnc: Quantifying global token attribution by incorporating the whole encoder layer in transformers. *arXiv preprint arXiv:2205.03286* (2022).
[28] Besmira Nushi, Ece Kamar, Eric Horvitz, and Donald Kossmann. 2017. On human intellect and machine failures: Troubleshooting integrative machine learning systems. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. 1017–1025.
[29] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an asr corpus based on public domain audio books. In *Proceedings of the IEEE international conference on acoustics, speech and signal processing*. 5206–5210.
[30] Su Pang, Daniel Morris, and Hayder Radha. 2020. CLOCs: Camera-LiDAR object candidates fusion for 3D object detection. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 10386–10393.
[31] Md. Rizwan Parvez, Wasi Uddin Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2021. Retrieval Augmented Code Generation and Summarization. In *Proceedings of the Findings of the Association for Computational Linguistics*. 2719–2734.
[32] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 1–18.
[33] Zi Peng, Jinqiu Yang, Tse-Hsun (Peter) Chen, and Lei Ma. 2020. A First Look at the Integration of Machine Learning Models in Complex Autonomous Driving Systems: A Case Study on Apollo. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1240–1250.
[34] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *Proceedings of the International conference on machine learning*. 28492–28518.
[35] Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. *arXiv preprint arXiv:1806.03822* (2018).
[36] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).
[37] Xuezhi Song, Yun Lin, Yijian Wu, Yifan Zhang, Siang Hwee Ng, Xin Peng, Jin Song Dong, and Hong Mei. 2022. RegMiner: Mining Replicable Regression Dataset from Code Repositories. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1711–1715.
[38] Megha Srivastava, Besmira Nushi, Ece Kamar, Shital Shah, and Eric Horvitz. 2020. An Empirical Analysis of Backward Compatibility in Machine Learning Systems. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3272–3280.
[39] Dan Su and Pascale Fung. 2020. Improving Spoken Question Answering Using Contextualized Word Representation. In *Proceedings of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing*. 8004–8008.
[40] Ryuichi Takanobu, Qi Zhu, Jinchao Li, Baolin Peng, Jianfeng Gao, and Minlie Huang. 2020. Is Your Goal-Oriented Dialog Model Performing Really Well? Empirical Analysis of System-wise Evaluation. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 297–310.
[41] Ze Tang, Jidong Ge, Shangqing Liu, Tingwei Zhu, Tongtong Xu, Liguo Huang, and Bin Luo. 2023. Domain Adaptive Code Completion via Language Models and Decoupled Domain Databases. In *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering*. 421–433.
[42] Hugging Face Team. 2022. *Hugging Face Website.* Retrieved September 8, 2024 from https://huggingface.co/
[43] Hugging Face Team. 2024. *Multinomial Sampling.* Retrieved September 12, 2024 from https://huggingface.co/docs/transformers/generation_strategies#multinomial-sampling
[44] Hugging Face Team. 2024. *Transformers Decoding Strategies.* Retrieved September 8, 2024 from https://huggingface.co/docs/transformers/main/en/generation_strategies#decoding-strategies
[45] LangChain Team. 2024. *Caching Embeddings.* Retrieved September 12, 2024 from https://python.langchain.com/v0.1/docs/modules/data_connection/text_embedding/caching_embeddings/

[46] LangChain Team. 2024. *Contextual Compression.* Retrieved September 12, 2024 from https://python.langchain.com/v0.1/docs/modules/data_connection/retrievers/contextual_compression/
[47] YOLO Team. 2022. *YOLO-v5.* Retrieved February 14, 2025 from https://zenodo.org/records/7347926
[48] Shogo Tokui, Susumu Tokumoto, Akihito Yoshii, Fuyuki Ishikawa, Takao Nakagawa, Kazuki Munakata, and Shinji Kikuchi. 2022. Neurecover: Regression-controlled repair of deep neural networks with training history. In *Proceedings of the IEEE International Conference on Software Analysis, Evolution and Reengineering*. 1111–1121.
[49] JC Torres. 2022. *Galaxy S10 5G update reportedly breaks face recognition.* Retrieved September 8, 2024 from https://www.slashgear.com/galaxy-s10-5g-update-reportedly-breaks-face-recognition-26684006/
[50] Frederik Träuble, Julius von Kügelgen, Matthäus Kleindessner, Francesco Locatello, Bernhard Schölkopf, and Peter Gehler. 2021. Backward-Compatible Prediction Updates: A Probabilistic Approach. In *Proceedings of the Advances in Neural Information Processing Systems*. 116–128.
[51] Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024. Knowledge Fusion of Large Language Models. In *Proceedings of the Twelfth International Conference on Learning Representations*.
[52] Changhan Wang, Yun Tang, Xutai Ma, Anne Wu, Sravya Popuri, Dmytro Okhonko, and Juan Pino. 2020. Fairseq S2T: Fast speech-to-text modeling with fairseq. *arXiv preprint arXiv:2010.05171* (2020).
[53] Haijun Wang, Yun Lin, Zijiang Yang, Jun Sun, Yang Liu, Jinsong Dong, Qinghua Zheng, and Ting Liu. 2019. Explaining regressions via alignment slicing and mending. *IEEE Transactions on Software Engineering* 47, 11 (2019), 2421–2437.
[54] Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. *arXiv preprint arXiv:1905.08743* (2019).
[55] Ruihan Wu, Chuan Guo, Awni Y. Hannun, and Laurens van der Maaten. 2021. Fixes That Fail: Self-Defeating Improvements in Machine-Learning Systems. In *Proceedings of the 35th Conference on Neural Information Processing Systems*. 11745–11756.
[56] Yijing Wu, SaiKrishna Rallabandi, Ravisutha Srinivasamurthy, Parag Pravin Dakle, Alolika Gon, and Preethi Raghavan. 2023. HeySQuAD: A Spoken Question Answering Dataset. *arXiv preprint arXiv:2304.13689* (2023).
[57] Yuqing Xie, Yi-An Lai, Yuanjun Xiong, Yi Zhang, and Stefano Soatto. 2021. Regression Bugs Are In Your Model! Measuring, Reducing and Analyzing Regressions In NLP Model Updates. *arXiv preprint arXiv:2105.03048* (2021).
[58] Sijie Yan, Yuanjun Xiong, Kaustav Kundu, Shuo Yang, Siqi Deng, Meng Wang, Wei Xia, and Stefano Soatto. 2021. Positive-congruent training: Towards regression-free model updates. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14299–14308.
[59] Yan Yan, Yuxing Mao, and Bo Li. 2018. Second: Sparsely embedded convolutional detection. *Sensors* 18, 10 (2018), 3337.
[60] Shin Yoo and Mark Harman. 2012. Regression Testing Minimization, Selection and Prioritization: A Survey. *Software Testing, Verification and Reliability* 22, 2 (2012), 67–120.
[61] Chenyu You, Nuo Chen, and Yuexian Zou. 2021. Knowledge Distillation for Improved Accuracy in Spoken Question Answering. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal*. 7793–7797.
[62] Hanmo You, Zan Wang, Junjie Chen, Shuang Liu, and Shuochuan Li. 2023. Regression fuzzing for deep learning systems. In *Proceedings of the IEEE/ACM 45th International Conference on Software Engineering*. 82–94.
[63] Jihu Zhang, Xiaochuan Jing, Wei Zhang, Haipeng Wang, and Yunwei Dong. 2016. Improve the Quality of ARC Systems Based on the Metamorphic Testing. In *Proceedings of the International Symposium on System and Software Reliability*. 137–141.
[64] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2022. Machine Learning Testing: Survey, Landscapes and Horizons. *IEEE Transactions on Software Engineering* 48, 1 (2022), 1–36.
[65] Qi Zhu, Zheng Zhang, Yan Fang, Xiang Li, Ryuichi Takanobu, Jinchao Li, Baolin Peng, Jianfeng Gao, Xiaoyan Zhu, and Minlie Huang. 2020. ConvLab-2: An Open-Source Toolkit for Building, Evaluating, and Diagnosing Dialogue Systems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 142–149.

Received 2024-09-13; accepted 2025-04-01