2025 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU) | 979-8-3315-4426-3/25/$31.00 ©2025 IEEE | DOI: 10.1109/ASRU65441.2025.11434593

Iterative Feedback in the Online Active Learning Paradigm

Mark Lindsey
Content Analytics Division
Probity Inc.
Herndon, USA
mlindsey@probity.com

Francis Kubala
Content Analytics Division
Probity Inc.
Herndon, USA
fkubala@probity.com

Richard M. Stern
Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, USA
rms@cmu.edu

***Abstract*—Online Active Learning with Corrective Feedback (OAL-CF) is a new machine learning paradigm that learns to detect events of interest in streaming audio by adapting to feedback from an operational user, who is a domain expert. The machine learns from each batch of the data stream by posing active learning queries to the expert and updating its model before making predictions. The expert reviews the predictions and indicates which of them are incorrect. This feedback is used to update the model for the next batch.**

In this paper, we introduce iterative feedback, where active learning and corrective feedback are performed multiple times per batch. We validate this approach on a large Spoken Language Verification task with 35 low-prevalence languages. The evaluation metric used (IMLM) accounts for the total cost of the method (i.e., error and feedback cost combined). The iterative algorithm achieves a 47.7% relative reduction in total cost compared to the original OAL-CF algorithm.

***Index Terms*—online machine learning, human-in-the-loop, active learning, corrective feedback**

I. Introduction

Online Active Learning (OAL) [1] and Online Active Learning with Corrective Feedback (OAL-CF) [2] are human-in-the-loop Machine Learning (ML) paradigms that are designed to learn to detect events of interest in streaming media. These paradigms have been shown to be both more accurate and less labor-intensive for processing data streams compared to Passive Machine Learning (PML) paradigms, specifically for tasks involving streaming audio.

These two paradigms are particularly effective at detecting rare events of interest in unique operating environments for two primary reasons. First, they learn directly from the operating environment, avoiding the assumption that the distribution of an outside dataset will match the distribution of the data being evaluated. Second, they are guided by feedback from a human domain expert.

In the case of OAL, the feedback channel takes the form of the expert providing labels for a few Active Learning (AL) queries selected by the machine from the current batch. These labels can be used to update the classifier and improve performance on the rest of the current batch. The basic process for the OAL algorithm begins by initializing the classifier with a small number of bootstrap examples representing both target and non-target data. Then, the algorithm begins the adaptation cycle. This is where the machine loads a batch a data from the stream, poses AL queries to the expert, updates its classification model based on the expert's responses, classifies the remaining unlabeled samples in the batch, then moves on to the next batch in the stream.

In addition to the AL channel, OAL-CF also allows the expert to provide Corrective Feedback (CF) for any errors observed in the classifier's predictions. As shown in Fig. 1, the OAL-CF pipeline contains the same steps as OAL (posing queries, updating the model, making predictions), plus the added steps of receiving CF from the expert and updating the model again before moving to the next batch. It is assumed here that the expert expects useful information to come only from samples predicted by the classifier to contain the target, and so only these positive predictions (composed of true and false positives) receive labels in the CF process.

Funding provided by Probity Inc.

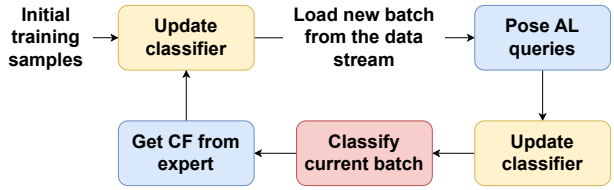

Fig. 1. The OAL-CF paradigm pipeline for classifying and learning from data streams. For each batch of data, the system poses one set of AL queries and receives one round of CF before loading the next batch.

For both OAL and OAL-CF, the goal of the algorithm is to achieve joint minimization of the error rate and annotation effort required of the expert. This joint minimization is measured by the Interactive Machine Learning Metric (IMLM) [3], which is used to evaluate these tasks.

In the original definition of these two paradigms, feedback from the AL and CF channels only happens one time for every batch of data. Furthermore, the CF samples obtained in the OAL-CF pipeline are only applied to batches later in the stream since the model only makes predictions once per batch. These restrictions are in place to prevent the annotation cost from growing too large.

In this paper, we remove these restrictions and formally introduce the Online Active Learning with Iterative Feedback (OAL-IF) ML paradigm, where multiple rounds of feedback and model updates can be applied iteratively to the same batch of data. Iterative Feedback (IF) gives the OAL-IF algorithm the opportunity to discover and correct False Negative (FN) errors (i.e., missed detections of the target), which are assumed to carry a high relative cost to the detection operation when the target is rare.

The potential disadvantage of IF is the possibility of incurring significantly more False Positive (FP) errors or raising the annotation cost in the process of recovering FNs. As such, determining how many times the algorithm should iterate and budgeting the appropriate amount of AL queries are key factors in successfully implementing OAL-IF. Other work has focused on AL budget allocation for the OAL paradigm, including budgets based on concept drift, or changes in the underlying data distribution over time [4]. For example, budget-constrained [5] and unconstrained [6] dynamic AL query allocation has been implemented based on supervised [7] and unsupervised

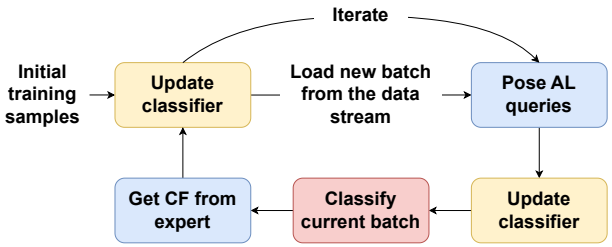

Fig. 2. The OAL-IF paradigm pipeline where iteration within the feedback loop is allowed before moving to the next batch. Compare to the OAL-CF loop (Fig. 1), which does not allow for iteration.

drift detectors [8]. These methods allow more queries to be made in batches where drift is detected, while fewer queries are made in stationary batches. In this paper, we introduce some simple stopping criteria and AL budgeting methods specific to the OAL-IF paradigm.

We validate the OAL-IF paradigm by applying it to the Spoken Language Verification (SLV) task and comparing the results to OAL and OAL-CF. While these OAL-based paradigms can be applied to any data stream, the data stream used in this paper contains target classes of extremely low prevalence. This choice was made intentionally because 1) many real-world operations perform detection on highly imbalanced data streams, 2) low target prevalence makes the task challenging and allows for analysis of algorithmic performance in these extreme cases, and 3) data streams with extreme class imbalance are rarely investigated in the machine learning literature.

In summary, this paper describes the following contributions.

1) Introduction of the OAL-IF paradigm
2) Basic stopping criteria for IF
3) Simple AL budgeting methods for OAL-IF
4) Experimentation with OAL-based paradigms on data streams with extremely low target prevalence

The remainder of this paper is organized as follows. Section II specifies the technical details of our contributions. Section III defines the experiments used to validate our technical contributions. Section IV reports and analyzes the results of the experiments. Finally, conclusions are discussed in Section V.

II. Methodology

This section details the technical contributions of this work. First, we describe how IF is implemented in the OAL-IF paradigm, including the classification pipeline and the iteration stopping criteria. We subsequently outline methods for limiting the active query budget in the IF setting.

A. Iterative Feedback

Extending the OAL-CF paradigm to include IF is simple but impactful. Just as in the OAL-CF paradigm, OAL-IF training begins by initializing the classifier with a small, class-balanced bootstrap corpus—equal parts from the target class and the non-target class. The system then processes a batch of samples from the data stream by posing AL queries, updating the classifier, making predictions, receiving CF from the expert, and updating again.

OAL-IF differs from OAL-CF in that rather than immediately moving on to the next batch (as would be done in the OAL-CF system) the OAL-IF system has the opportunity to repeat the query, prediction, feedback, and model update steps until some stopping criterion is reached. This within-batch iterating procedure is illustrated in Fig. 2. OAL-IF can also be seen as a generalization of OAL-CF. Conversely, OAL-CF can be considered a specific configuration of OAL-IF that is defined to always iterate through these steps one time per batch.

The purpose of repeating these steps is to ensure that target samples are not missed. FNs are assumed to be much more detrimental to the operation of validation and detection tasks than FPs because the targets are so rare. Hence, giving the system extra chances to eliminate FN errors via IF is worthwhile, as long as the total cost of extra annotation and additional FP errors is less than OAL-CF. The trade-off between decreased FNs and increased FPs plus increased annotation cost can be measured using IMLM (see Section III-D) and optimized by adjusting the configuration of the system.

Stopping Criteria: The choice of the criterion for stopping the iteration cycle is a crucial factor for optimizing the trade-off between annotation cost and error rate. For simplicity, we only implement two basic rule-based stopping criteria in this work. The two criteria are:

1) **Fixed Iterations (*FI*)**: Stop after iterating a specified number of times.
2) **No True Positives Detected (*NTPD*)**: Stop when no more true positive predictions are confirmed by CF.

The *FI* criterion allows for controlled experimentation. The *NTPD* criterion is based on the hypothesis that CF does not significantly improve results once it has discovered all the target samples.

B. Limiting the Active Query Budget

One component of the OAL-CF algorithm that needs to be adjusted with the addition of IF is the active query budget, which defines how many AL queries can be made per batch. AL is repeated along with CF in the IF loop and serves as a crucial method for identifying concept drift that might go undetected by CF. In the baseline OAL and OAL-CF algorithms, AL is restricted to 8 queries per batch. However, in the OAL-IF paradigm, AL can quickly become costly if the number of queries per batch experiences a significant increase with additional iterations. For example, if 8 AL queries are allowed per iteration and there are two iterations per batch, the annotation cost of AL doubles to 16.

As such, we introduce two simple methods to reduce the active query budget while retaining reasonable error rates.

1) Iterating Corrective Feedback Only: One way to restrict the AL budget in the IF setting is to allow only one iteration of AL queries per batch but allow multiple iterations of CF. This way, the AL budget is guaranteed to be the same as the baseline OAL-CF configuration while the CF restrictions are loosened. In addition to restricting the AL budget, this method is also useful for direct analytical comparison to the baseline.

This method is referred to as "CF Only" (*CFO*) in the experiments below.

2) Limiting Non-target Active Queries: Another budgeting approach is to limit the number of queries collected when the classifier predicts that a batch contains no target samples. So, before the AL queries are made, the model makes predictions for each sample in the batch. If none of these predictions are target predictions, then fewer AL queries are posed to the expert than would be posed otherwise. The assumption that motivates this approach is that target labels are more valuable on average than non-target labels because of the relative importance and scarcity of the target class.

For this method, rather than limiting the number of queries for putative non-target batches to 0, we set the limit to 2 queries. Preliminary trials showed that completely eliminating AL queries in putative non-target batches allowed for too many target samples to go undetected, so the limit was set to 2 queries as a precautionary measure.

This method is referred to as "Non-target Limiting" (*NTL*) in the experiments below.

III. Experimental Configuration

The experiments outlined in this section compare OAL-IF with the baseline one-iteration version of OAL and OAL-CF. The configurations of OAL-IF in the experiment include all possible combinations of stopping criteria (*FI* and *NTPD*) and budget-limiting methods (*no limit*, *CFO*, and *NTL*) described in Section II. The following subsections describe the task, data, classifier, and evaluation methodology used in these experiments.

A. Task and Corpus

The SLV task is to determine whether an audio recording contains a given target language. This differs from the Language Identification (LID) task, which is to predict which language from a list of languages is being spoken in an audio recording. It is also different from language detection, which involves detecting a target language in a data stream when there is potential overlap with other languages.

The data for the SLV task in this work come from Mozilla's Common Voice Corpus [9]. The Common Voice Corpus contains short recordings (approximately 5 seconds each, on average) of speech audio. Each recording is a single utterance from one speaker in only one language. This means that no samples contain overlapping speech or multiple language classes, much like an Automatic Speaker Verification (ASV) scenario. Language labels are extracted from the metadata provided with the Corpus.

From this Corpus, we form four subcorpora with languages from the four following geographic regions: the African continent, the Caucasus region, South Asia, and Southeast Asia. We chose these four regions for their linguistic diversity, and so that similar, confusable languages would appear alongside each other to make the verification problem more representative of real localized collection operations. The data we used for evaluation come from the validation and test splits of these languages, which doubles the size of the evaluation data. This differs from our previous work that uses Common Voice for SLV that only utilizes the test split [2], [3]. The subcorpus durations and languages are listed in Table I.

Region	Africa	Caucasus	SE Asia	S Asia
Duration	136 hours	284 hours	226 hours	169 hours
Language	**Afrikaans** **Amharic** **Basaa** **Dioula** **Hausa** Igbo Kabyle Kinyarwanda Luganda Swahili **Tamazight** **Tigre** Tigrinya Twi **Yoruba**	Abkhaz **Armenian** Azerbaijani Bashkir Belarusian **Central Kurdish** **Chuvash** **Erzya** Georgian **Kazakh** **Kurmanji Kurdish** **Kyrgyz** **Moksha** Ossetian Russian **Serbian** **Tatar** Turkish Turkmen Ukrainian Uzbek	Arabic **Cantonese** Mandarin English French **Indonesian** Lao Portuguese Tamil Thai **Vietnamese**	**Assamese** Bengali **Hakha Chin** **Dhivehi** English Persian **Hindi** **Malayalam** **Marathi** **Nepali** **Odia** **Punjabi** **Pashto** **Santali** **Saraiki** Tamil **Telugu** **Turkmen** Urdu Uzbek

TABLE I
Languages and durations for the four Common Voice subcorpora. Target languages are bolded.

Of the languages in these subcorpora, we select the languages of low prevalence in each subcorpus as the validation targets, where each chosen language is present in less than 5% of samples. This differs from our previous SLV work that considers languages with prevalence between 1% and 5% as targets, but not less than 1%. These low-prevalence languages are chosen to make the task more difficult, and to simulate the reality of detecting extremely scarce targets in real-world operations. The resulting 35 target languages, their abbreviations, source corpora, and overall prevalence are listed in Table II.

Language	Abbr.	Corpus	% of data stream
Afrikaans	af	Africa	0.11%
Amharic	am		0.42%
Basaa	bas		0.92%
Dioula	dyu		0.10%
Hausa	ha		1.14%
Tamazight	zgh		0.15%
Tigre	tig		0.25%
Yoruba	yo		1.71%
Armenian	hy-AM	Caucasus	3.74%
Central Kurdish	ckb		4.63%
Chuvash	cv		1.12%
Erzya	myv		0.32%
Kazakh	kk		0.45%
Kurmanji Kurdish	kmr		3.43%
Kyrgyz	ky		1.42%
Moksha	mdf		0.07%
Serbian	sr		1.58%
Tatar	tt		3.81%
Cantonese	yue	SE Asia	2.89%
Indonesian	id		3.78%
Vietnamese	vi		1.06%
Assamese	as	S Asia	0.79%
Hakha Chin	cnh		1.13%
Dhivehi	dv		3.27%
Hindi	hi		4.13%
Malayam	ml		1.09%
Marathi	mr		2.61%
Nepali	ne-NP		0.31%
Odia	or		0.51%
Punjabi	pa-IN		0.57%
Pashto	ps		0.15%
Santali	sat		0.11%
Saraiki	skr		1.58%
Telugu	te		0.07%
Turkmen	tk		0.40%

TABLE II
Prevalence of all target languages in the SLV corpora.

Since the Common Voice Corpus was collected asynchronously, there is no chronological order to the data. As such, the data need to be organized in some way in order to form a data stream. To do this, we randomly assign the order in which the samples are presented according to a Dirichlet distribution, which is common practice in the Test-Time Adaptation (TTA) literature [10]. One random Dirichlet ordering is generated per subcorpus. These orderings are used for all experiments for consistency.

B. Classifier Architecture

The classification model used here (see Fig. 3) is the same contrastive classifier neural network used in our previous AL and OAL work [11]. This classifier is composed of a feature extractor and two feed-forward networks: the contrastive Siamese network and the classifier network.

The feature extractor is a pre-trained ECAPA-TDNN embedding model trained on the LID task [12]. The pre-trained embedding

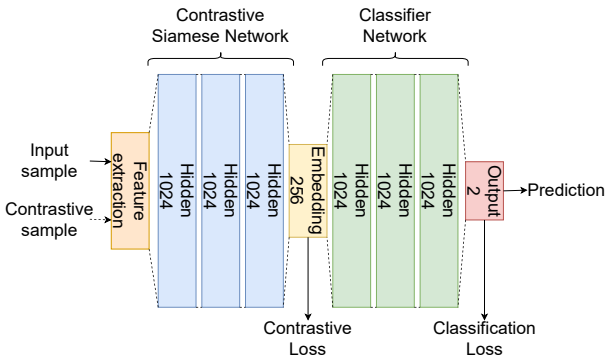

Fig. 3. The contrastive classifier architecture used for the experiments. The classifier consists of a pre-trained feature extractor and two feed-forward networks for contrastive modeling and classification.

model comes from SpeechBrain[1]. The parameters of this model are frozen, so the feature extractor is not optimized during training. Each embedding feature represents one sample (i.e., one audio recording from the Common Voice Corpus) with no overlap from other samples. As dictated by the design of the ECAPA-TDNN model, each audio recording is compressed into a 256-dimensional single-vector representation. The input to the feed-forward networks is formed by concatenating the embedding from the sample being classified with the embeddings from the three samples immediately preceding and the three samples immediately following the sample to be classified. This concatenated feature, which now has 1,792 dimensions, gives the model context, which was found to improve performance.

The contrastive network separates the classes using a contrastive loss [13] and compresses the features into a smaller embedding space with 256 dimensions. The classifier network optimizes a class-weighted cross-entropy loss, where the class weights are inversely proportional to the class prevalence. This is referred to as Inverse Class Frequency (ICF) weighting [14]. ICF weighting can be calculated using Eq. 1, where w_y is the weight of class y, C is the number of classes, N is the total number of samples in the adaptation pool, and N_y is the number of samples from class y in the adaptation pool. The ICF weighting is updated every time the adaptation pool is updated.

$$w_y = \frac{N}{C \cdot N_y} \tag{1}$$

The contrastive loss, $\mathcal{L}_{\text{contrast}}$, and cross-entropy loss, $\mathcal{L}_{\text{class}}$, are combined with weighting factor ψ, resulting in the total loss function shown in Eq. 2.

$$\mathcal{L} = \psi \mathcal{L}_{\text{class}} + \mathcal{L}_{\text{contrast}} \tag{2}$$

C. Learning Parameters

The learning parameter settings for all experiments are identical, where possible, for comparability. The most important shared learning parameters are listed in Table III.

Furthermore, we employ uncertainty sampling [15] based on the output of the Softmax layer of the classifier [16] as the AL query selection strategy for all experiments. All model updates are made by fine-tuning the network.

[1]https://huggingface.co/speechbrain/lang-id-voxlingua107-ecapa

Parameter(s)	Value/Setting
Optimizer	Adam
Learning rate	10^{-4}
Weight decay	10^{-5}
Training batch size	256
Early stopping patience	5 epochs
Early stopping margin	1e-3
Bootstrap corpus size	8 samples
Stream batch size	800 samples

TABLE III
THE SHARED LEARNING PARAMETERS FOR ALL EXPERIMENTS.

D. Evaluation Methodology

We employ the Interactive Machine Learning Metric (IMLM) [3] as the primary evaluation metric in this paper. IMLM represents the relative reduction in total cost that an algorithm achieves compared to manual inspection of every sample in the data stream. Here, total cost is measured in time, where errors (both FNs and FPs) and annotations cost the expert some amount of time. This time cost is then divided by the total cost of manual inspection to form a ratio. The formula used to calculate IMLM is shown in Eq. 3, where N_{ann} is the number of bootstrap corpus samples plus the number of AL queries, N_{fp} is the number of FPs, N_{fn} is the number of FNs, and N_{target} is the number of target samples in the data stream. Since IMLM is a ratio, results less than 1 indicate a decrease in time cost when using the algorithm, and results greater than 1 indicate increased time cost.

$$\text{IMLM} = \frac{N_{\text{ann}} + N_{\text{fp}}}{N} + \frac{N_{\text{fn}}}{N_{\text{target}}}. \tag{3}$$

As a secondary evaluation metric, we use the Detection Cost Function (DCF) [17], as defined in Eq. 4. DCF is a linear combination of the FP (P_{fp}) and FN (P_{fn}) error rates. The weighting factor is set to $\xi = 0.75$, meaning that FNs are weighted three times as much as FPs. This weighting is intentional, as FNs are more detrimental to the operation than FPs. This weighting is also consistent with numerous other evaluations that employ the DCF metric [18]–[20].

$$\text{DCF} = (1 - \xi) P_{\text{fp}} + \xi P_{\text{fn}} \tag{4}$$

We use a modified form of postquential analysis [3] to track the evaluation metrics over time. In this case, postquential analysis means that we measure the performance of the system by the cumulative errors and annotations accrued after the last iteration of each batch. In order to track the total time cost for the expert across the whole experiment, the FP and annotation counts include all FPs and AL queries accrued at the end of each iteration. The FN count is simply all remaining undetected target samples at the end of the last iteration.

For further analysis, we also track the performance of the system before and after the AL update of every iteration in each batch. This allows us to identify how much impact each iteration has on the overall performance.

IV. Results and Analysis

The cumulative experimental results across all languages and all subcorpora are listed in Table IV. These results highlight very clearly the advantage of using OAL-IF over either of the baselines. Only the worst IMLM score from OAL-IF is equal to that of OAL-CF, all other scores being lower. The DCF scores are all much lower than the baselines.

The best configuration in terms of IMLM is OAL-IF with the AL budget limited for putative non-target batches and three iterations

Paradigm	AL limit	Stop Crit.	IMLM	DCF
OAL	8	*FI*-1	0.093	0.058
OAL-CF	8	*FI*-1	0.044	0.023
OAL-IF	*No limit*	*FI*-2	0.035	0.011
		FI-3	0.041	0.009
		FI-4	0.044	***0.007***
		NTPD	*0.030*	0.013
	CFO	*FI*-2	0.031	0.013
		FI-3	0.030	0.012
		FI-4	*0.028*	*0.011*
		NTPD	0.029	0.012
	NTL	*FI*-2	0.024	0.011
		FI-3	***0.023***	*0.008*
		FI-4	0.026	*0.008*
		NTPD	0.031	0.017

TABLE IV
CUMULATIVE RESULTS FOR THE BASELINES AND ALL CONFIGURATIONS OF OAL-IF. *FI*-N REFERS TO RUNS WITH N FIXED ITERATIONS, *NTPD* STOPS AFTER CF DISCOVERS NO NEW TARGET SAMPLES, AND *CFO* AND *NTL* ARE THE TWO AL BUDGETING METHODS.

per batch. This run was only marginally better than the other runs using the same AL budgeting method, which contributed the top three IMLM scores in the table.

The *NTPD* stopping criterion did not successfully reduce the IMLM compared to the *FI* methods because of an elevated FN count. This indicates that sometimes there may be a set of target samples present in a batch that goes completely undetected without additional iterations.

The best system in terms of DCF was the OAL-IF system with no AL budget and four iterations per batch. Interestingly, this was also the OAL-IF run with the highest IMLM. This indicates that the low DCF was made possible at the cost of extra annotations in the form of AL queries. The configurations that achieved lower IMLM scores did so by achieving a marginally higher DCF while incurring a much lower annotation cost.

Considering the best scores for each paradigm, we see that progressing from OAL to OAL-CF results in approximately a 50% reduction of both IMLM and DCF. Then, progressing from OAL-CF to the best OAL-IF configuration here results in another reduction of approximately 50%.

We perform a Wilcoxon signed-rank test [21] across the scores of all 35 individual languages to determine whether the improvement from OAL-CF to OAL-IF is statistically significant. The test indicates that OAL-IF is better than OAL-CF with over 99% confidence, as the resulting p-values are 2.3×10^{-6} for IMLM and 7.1×10^{-6} for DCF.

A. Prevalence and Individual Language Results

Observing the behavior of smaller groups of languages and individual languages in addition to the overall results provides an important perspective. By observing the difference in IMLM scores between the lowest prevalence languages (containing $< 1\%$ target) and the higher prevalence languages ($> 1\%$, $< 5\%$ target) in Fig. 4, we see that the lower prevalence languages are much more difficult to detect effectively. This is to be expected, as an extremely rare target can make it difficult to provide enough examples for the classifier to learn the target distribution. One other piece of evidence to support this claim is that the only language to score better using OAL-CF than using the best configuration of OAL-IF is Telugu, which has an extremely low target prevalence of 0.07%.

We also see in Fig. 4 that the optimal number of iterations is clearly different for the low and high prevalence languages. High

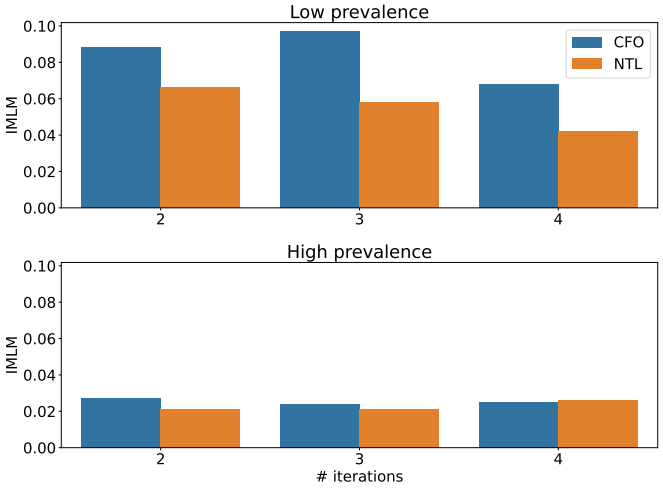

Fig. 4. IMLM scores split into the low prevalence ($< 1\%$ target) and higher prevalence ($> 1\%$, $< 5\%$ target) for varying numbers of iterations.

prevalence languages achieve the lowest IMLM at 3 iterations, while low prevalence continue to see a decrease in IMLM when 4 iterations are used. This behavior highlights the need for a flexible AL budget that adapts automatically to an estimate of the target prevalence.

B. Trend Analysis

The addition of IF has an interesting effect on the IMLM scores over time. Figs. 5 and 6 show cumulative metrics over time, which means that each point on the curve represents the indicated metric calculated based on all errors and annotations for all languages up to and including the batch indicated on the horizontal axis. Since some subcorpora are shorter than others, the languages from the shorter subcorpora only contribute errors and annotations to the cumulative metrics until their last batch.

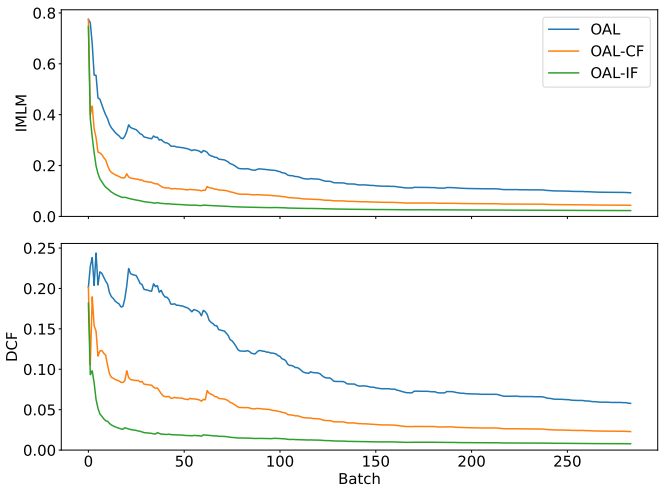

Fig. 5. The postquential trends of the OAL, OAL-CF, and OAL-IF paradigms for IMLM and DCF. The OAL-IF curves are always lower and smoother than the baseline curves.

Fig. 5 shows the postquential trend, or the cumulative scores from the classifier after the last update for each batch. As expected based on the overall scores from Table IV, the OAL-IF curve is always lower than either of the baseline curves for both IMLM and DCF.

The additional information we glean from these curves is that OAL-IF decreases very quickly in the early batches, and that it is a relatively smooth curve. The early decrease is an indication that IF helps the classifier learn the task much faster. The smoothness of the curve indicates that IF yields more stable learning that is less affected by new information or concept drift in the data stream.

Fig. 6 shows the IMLM and DCF trends before iterating and after each iteration in a 4-iteration configuration of OAL-IF. We observe a large gap between the pre-iteration curve and the first iteration curve, then progressively smaller improvements after each iteration. This is a reasonable behavior, as a large number of iterations eventually converge to the best possible error rate for the system. We also observe that the curves grow increasingly smooth as iterations progress. This indicates that more iterations are likely to recover more errors in the early batches and make learning more stable.

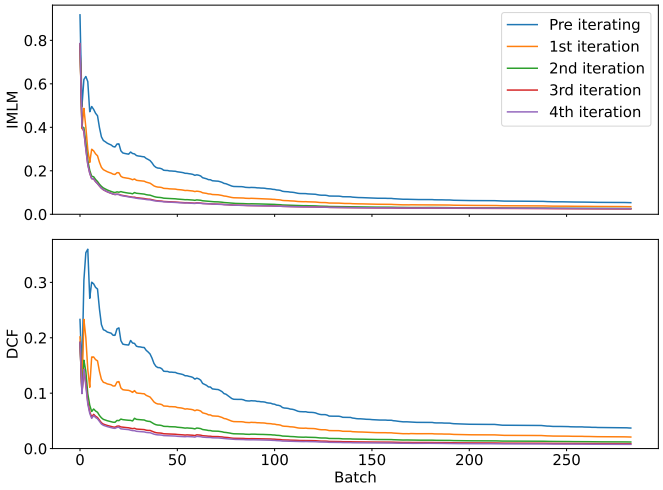

Fig. 6. The trends for IMLM and DCF before IF and after each iteration in the OAL-IF paradigm. Improvements shrink as the iterations increase.

V. Conclusions

In this work, we introduced Online Active Learning with Iterative Feedback (OAL-IF), an extension of Online Learning with Corrective Feedback (OAL-CF) that allows for multiple iterations of active queries and corrective feedback for each batch of samples from the data stream. We also introduced methods for limiting the active query budget to keep the annotation cost of OAL-IF low. Experimental results indicate that OAL-IF makes a worthwhile trade-off between additional annotations and reduced false negative errors, with a significant overall reduction in IMLM of 47.7% relative to the baseline OAL-CF system. Additional analysis shows that OAL-IF performs well when targets are extremely scarce, and that the addition of IF makes learning faster and more stable.

Our analysis of the experimental results also reveals two clear directions for future work. First, the stopping criteria and AL budgeting methods introduced in this paper had significant impact on the overall performance of the system, but they were perhaps overly simplistic. More flexible stopping criteria and more adaptive AL budgeting are key to further minimization of the total cost of the algorithm. Second, there still remains one language where OAL-IF achieved a worse score than the original OAL baseline. Further development of these systems is necessary to guarantee that the scores will always be better than the OAL scores.

References

[1] D. Cacciarelli and M. Kulahci, "Active Learning for Data Streams: a Survey," *Machine Learning*, vol. 113, no. 1, pp. 185–239, 2024.

[2] M. Lindsey, "Online Active Learning with Corrective Feedback: A New Paradigm Applied to Streaming Audio," Ph.D. dissertation, Carnegie Mellon University, 2024.

[3] M. Lindsey, F. Kubala, and R. M. Stern, "A Unified Metric for Simultaneous Evaluation of Error Rate and Annotation Cost," in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025.

[4] N. Lu, G. Zhang, and J. Lu, "Concept Drift Detection via Competence Models," *Artificial Intelligence*, vol. 209, pp. 11–28, 2014.

[5] G. J. Aguiar and A. Cano, "Dynamic Budget Allocation for Sparsely Labeled Drifting Data Streams," *Information Sciences*, vol. 654, p. 119821, 2024.

[6] Z. Wu, H. Wang, J. Guo, Q. Yang, and J. Shao, "Learning Evolving Prototypes for Imbalanced Data Stream Classification with Limited Labels," *Information Sciences*, p. 120979, 2024.

[7] A. Bifet and R. Gavalda, "Learning from Time-changing Data with Adaptive Windowing," in *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 2007, pp. 443–448.

[8] G. Ditzler and R. Polikar, "Hellinger Distance Based Drift Detection for Nonstationary Environments," in *2011 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*, 2011, pp. 41–48.

[9] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common Voice: A Massively-multilingual Speech Corpus," *arXiv preprint arXiv:1912.06670*, 2019.

[10] L. Yuan, B. Xie, and S. Li, "Robust Test-time Adaptation in Dynamic Scenarios," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 15 922–15 932.

[11] M. Lindsey, N. R. Robinson, F. Kubala, and R. M. Stern, "Reducing the Cost of Spoof Detection Labeling using Mixed-Strategy Active Learning and Pretrained Models," in *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2023, pp. 1–7.

[12] C. M, A. Mandal, and S. Mukherjee, "Study of ECAPA-TDNN Models for Spoken Language Identification Task," in *2023 IEEE World Conference on Applied Intelligence and Computing (AIC)*, 2023, pp. 233–237.

[13] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality Reduction by Learning an Invariant Mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 2006, pp. 1735–1742.

[14] G. King and L. Zeng, "Logistic Regression in Rare Events Data," *Political analysis*, vol. 9, no. 2, pp. 137–163, 2001.

[15] D. D. Lewis, "A Sequential Algorithm for Training Text Classifiers: Corrigendum and Additional Data," in *Acm Sigir Forum*, vol. 29, no. 2. ACM New York, NY, USA, 1995, pp. 13–19.

[16] M. Sensoy, L. Kaplan, and M. Kandemir, "Evidential Deep Learning to Quantify Classification Uncertainty," *Advances in neural information processing systems*, vol. 31, 2018.

[17] F. Byers, J. Fiscus, Seyed, G. Sanders, and M. Przybocki, "Open Speech Analytic Technologies Pilot Evaluation OpenSAT Pilot," 2019-02-27 2019.

[18] D. Delgado, K. Walker, S. Strassel, K. S. Jones, C. Caruso, and D. Graff, "The SAFE-T Corpus: A New Resource for Simulated Public Safety Communications," in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, 2020, pp. 6450–6457.

[19] M. Lindsey, T. Vuong, and R. M. Stern, "Unsupervised Voice Type Discrimination Score Adaptation Using X-Vector Clusters," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[20] T. Vuong, Y. Xia, and R. M. Stern, "Learnable Spectro-temporal Receptive Fields for Robust Voice Type Discrimination," in *Interspeech 2020*. ISCA, Oct. 2020, pp. 1957–1961.

[21] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.