Error Amplification When Updating Deployed Machine Learning Models

George Alexandru Adam ALEX.ADAM@MAIL.UTORONTO.CA
University of Toronto, Vector Institute

Chun-Hao Kingsley Chang KINGSLEY@CS.TORONTO.EDU
University of Toronto, Vector Institute

Benjamin Haibe-Kains BENJAMIN.HAIBE.KAINS@UTORONTO.CA
University of Toronto, Vector Institute, University Health Network

Anna Goldenberg ANNA.GOLDENBERG@UTORONTO.CA
University of Toronto, Vector Institute, The Hospital for Sick Children

Abstract

As machine learning (ML) shows vast potential in real world applications, the number of deployed models has been increasing substantially, but little attention has been devoted to validating/improving model performance over time. Model updates, sometimes frequent, are essential to dealing with data shift, policy changes and in general, to improving the model performance. Updating also presents significant risks to amplifying model errors if effort is not put into preventing it. Unfortunately very little analysis is done to date of what can happen as models are deployed and become a part of the decision making process, where there is no longer a way to disentangle human from machine error in the collected labels. The phenomenon of interest is termed error amplification where model errors corrupt future labels, and are reinforced by updates eventually causing the model to predict its own outputs instead of the labels of interest. We analyze various factors influencing the magnitude of error amplification, and provide guidance for model and threshold selection when error amplification is a risk. We demonstrate that a variety of learning techniques cannot handle the systematic way in which error amplification corrupts observed outcomes. Additionally, we discuss both procedural and modeling solutions to reduce model deterioration over time based on our empirical evaluations.

1. Introduction

As applications of machine learning expand to include decision support systems that can directly impact the health and safety of humans such as autonomous vehicles and healthcare, it becomes imperative to focus on properties and updating regimes of these models that will ensure that they remain effective as time goes by. In healthcare, patient demographics shift over time, as do disease definitions and best practices, so having a static model that is trained once and never updated is often not a long-term option (Nestor et al., 2019). All models will have a tendency towards making errors depending on the application which dictates the desired tradeoff between false positive (FP) and false negative (FN) predictions. (Adam et al., 2020) give the example of a risk prediction model in the ICU that incorrectly suggests a patient is terminally ill (FP), at which point clinicians will place that patient

© 2022 G.A. Adam, C.-H.K. Chang, B. Haibe-Kains & A. Goldenberg.

on palliative care, potentially resulting in premature death. When such errors propagate into future labels and these labels are used for retraining/updating, the model will reinforce them, and deteriorate in performance over time, while achieving seemingly stable/improved performance according to observed labels. We refer to this cycle of predictions which can flip labels, and model updating as a **error amplification**. A precise definition is given in Section 3.

The results of our work are broadly applicable to deployed predictive models in healthcare, and in order to give practical grounding to the methods described throughout, we stick with the canonical ICU risk prediction task discussed by (Adam et al., 2020) due to its high-risk nature and the fact that similar models are currently deployed (Nestor et al., 2020). A common model validation paradigm prior to deploying an ML model in healthcare is the silent trial where a given model is evaluated prospectively without its predictions being revealed to clinicians. Unfortunately, silent trials are not able to capture error amplification, so we conduct simulations using retrospective MIMIC-IV data following the kind of retraining protocols that would be in place once a model is deployed.

We begin by formally defining error amplification, and examining several factors capable of influencing how quickly error amplification occurs. Then, we explore various ways of reducing error amplification including data valuation methods, dynamic sample reweighting, and other noisy label learning techniques. Our experiments reveal that basic heuristics are a surprisingly powerful baseline, and the implementation overhead of the other methods in some cases does not justify their use. We provide the following insights and practical advice:

Generalizable Insights about Machine Learning in the Context of Healthcare

1. How to choose classification thresholds to minimize the effect of error amplification while maintaining model utility. Naive threshold selection can significantly accelerate error amplification.
2. How to adjust model selection procedures when error amplification is a concern. Namely, a preference emerges for low-complexity, high-bias models which prevent the overfitting of model-induced label noise.
3. How to schedule update frequency to minimize error amplification.

2. Related Work

Error amplification has been discussed in prior works under the term feedback loop Adam et al. (2020), but the main focus has been on resource allocation Ensign et al. (2018) or recommender systems Mansoury et al. (2020), whereas we explore supervised learning. Sculley et al. discussed feedback loops arising as machine learning occupies more of the software stack, and makes predictions used by upstream systems without software engineers being aware of such an interaction and its repercussions. While slightly more subtle than the high-stakes data corruption theme that is at the core of our work, Sculley et al. stimulated more discussion of feedback loops in a wide variety of domains.

The first work that tried to actually characterize feedback loops looked at the effect of allocating policing resources based on a combination of discovered and reported crimes

(Ensign et al., 2018). It was proven theoretically that if resources are allocated only in proportion to the rate of discovered crime at different areas in a city, then eventually all resources would be sent exclusively to the area that had a slightly higher crime rate to begin with.

A line of application-specific work on feedback loops has analyzed how popularity bias can be worsened by recommender systems and cause reduced content diversity, shifting user preferences, and user homogenization (Mansoury et al., 2020). The recommender system setting enables an extreme version of feedback loops due to how often the model parameters are updated to take into account recent user interactions. Sinha et al. looked to decompose the effect of a recommender system on user-item ratings, from the underlying rating that would have occurred without the presence of a feedback loop. The authors found an increasingly stronger bias for datasets where data was accumulated over a period of time and there was a recommender system implemented from the beginning.

Performative prediction (Perdomo et al., 2020) is a relevant framework that closely captures the setting in our work. A performative prediction is defined as one that can influence the outcome it aims to predict. The optimization procedure called repeated risk minimization (RRM) was introduced to study the stability of model parameters under performative predictions. RRM would require explicitly identifying noisy samples and excluding them from updates which we show to be a difficult task, so it is not a direct solution to error amplification.

Several works have analyzed the real-world cases related to feedback loops. A real-world, though post-hoc analysis showing the existence of a feedback loop, was done in Malik (2020) where the Zestimate model used by the real estate website Zillow was shown to amplify housing price errors, particularly by inflating prices when overestimates are made. Khritankov focused on demonstrating how feedback loops can occur in general regression settings, but did not provide any advice on how to solve the problem. Lastly, Adam et al. (2020) focused on the ICU setting and binary classification where false positive predictions could propagate into future update data. The authors looked at basic heuristics for mitigating the feedback loop effect, but did not analyze the most important contributing factors or consider advanced regularization techniques like we do.

Other works have studied a simpler setting where label distributions (i.e. condition prevalence) change over time due to external causes not influenced by the model's predictions (Saerens et al., 2002; Lipton et al., 2018). Detecting label shift is an important feature that should be part of any model pipeline in deployment, but it is limited to suggesting when a model needs to be taken offline and does not solve the underlying problem. A detection-based solution to error amplification requires identifying specific samples which are mislabeled rather than detecting differences in proportion at the population level.

Lastly, Liley et al. (2021) use causal modelling to encapsulate the influence that machine predictions have on observed features and interventions. Their framework recognizes the role of the model in preventing outcomes as a constrained optimization problem where maximally intervening for each patient is not feasible due to limited resources. Their work theoretically analyzes the stability of models naively updated without taking into account their influence. Some of their proposed solutions are impractical requiring modelling of interventions, so we focus on practical modifications to the ML pipeline that can help stop error amplification.

3. Error Amplification Definition

Recall that error amplification is informally defined as model deterioration due to updating with data corrupted by the model's predictions. Let f_θ be the classifier parameterized by $\theta = \arg\min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{train}}} L(y, f_\theta(x))$ where $\mathcal{D}_{\text{train}}$ is the training data, L is a loss function, $x \in R^d$ are sample features, and $y \in \{0,1\}$ is the label. We assume that f_θ outputs $P(y=1|x)$ (predictive score from 0 to 1) and use $g_\theta^\tau = \mathbb{1}[f_\theta(x) \geq \tau]$ to denote the thresholded binary output with threshold τ. Error amplification can happen with as little as a single update, but can be more pronounced when a series of updates occur. This cycle of a classifier influencing data, which then influences the classifier is captured by the notion of repeated risk minimization analyzed by Perdomo et al.:

$$\theta_{t+1} = \arg\min_\theta \mathop{\mathbb{E}}_{(x,\hat{y})\sim\mathcal{D}_{\theta_t}} L(\hat{y}, f_\theta(x))$$

The notation $\mathcal{D}_{\theta_t}$ indicates that the current data distribution has been influenced by the model parameters at a previous time step. Unlike the examples discussed in the context of performative prediction where sample features x change, here the features x remain the same, and the ground truth label for incorrectly predicted samples changes from y to $\hat{y}$. To simplify our analysis, we assume only one kind of error can flip future labels, and focus on FPs, though the same results hold symmetrically if FNs are instead capable of flipping labels. This is a reasonable assumption since a critical care model predicting patient outcomes can only cause an outcome (death) if incorrectly deciding to give a critical patient palliative care rather than continuing treatment in ICU. We emphasize that EA is concerned with errors rather than correct predictions changing outcomes. In the latter case, a correct prediction (TP) results in an averted undesirable outcome, so the model appears to be wrong according to the observed label. This is also a risk for deployed models, but is fundamentally a different phenomenon which needs a different set of methods to be addressed, so it is not included as part of our EA definition. Thus, the data generating mechanism for labels becomes

$$\hat{y} = \begin{cases} 1 & \text{if } g_\theta^\tau(x) = 1 \land y = 0 \land z = 1 \\ 1 & \text{if } y = 1 \\ 0 & \text{otherwise} \end{cases}$$

where $z \sim \text{Ber}(p)$ is a Bernoulli random variable used to control the probability of an incorrect prediction becoming the observed label for that sample. Depending on the domain, p represents the fact that users might not always follow the model's predictions, or that even if an incorrect prediction is followed there is a chance that an outcome does not occur. We perform a worst-case analysis of $p = 1$ throughout representing fully autonomous decisions.

4. Factors Influencing Error Amplification

In this section we investigate 3 factors we deem to have a large impact on error amplification in order to provide ML teams with advice for robust model deployment. We stick with factors which are in the control of engineers, and we are not concerned with aspects of Human-AI interaction as we assume a worst-case full automation setting.

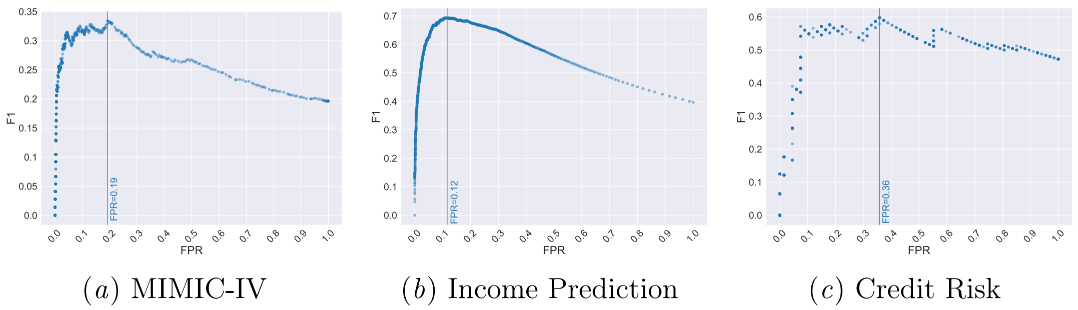


Figure 1: The relationship between F1 Score and FPR shown for 1000 linearly separated threshold values for our 3 datasets. Choosing a threshold that naively maximizes F1 Score index can lead to high false positive rates whereas accepting a slightly lower F1 Score can significantly lower the false positive rate. Vertical blue lines indicate the corresponding FPR for the maximal F1 Scores.

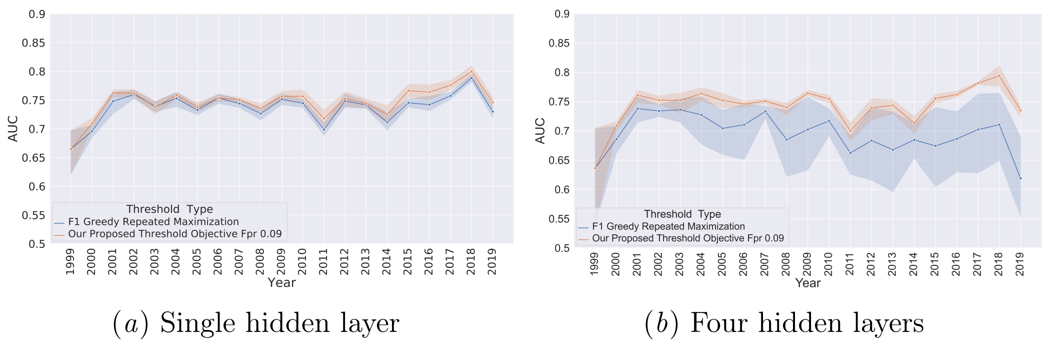


Figure 2: Comparison of our proposed threshold selection procedure and naive threshold selection on MIMIC-IV data. Naive maximization of F1 score results in a much larger FPR (0.19) than our proposed threshold selection approach (0.09) thus causing a worse model over time.

Experimental Details: Throughout the rest of the paper we use three different binary classification datasets (MIMIC-IV - healthcare dataset, Income data, Credit Risk Data) and 5 fully connected classification architectures. Most of our experiments focus on MIMIC-IV, and we use the other datasets to show the broader applicability of our results for datasets of various sizes and task difficulty. The task for MIMIC-IV is mortality prediction within 48 hours, and we bin temporal features and compute their mean and standard deviation per bin to simplify the modeling process. MIMIC-IV has a total of 31k samples and 48 features with along with a feature indicating the year of the patient's hospital stay. Unless otherwise mentioned, we treat data from 1996-1998 as training data, and update models on the current

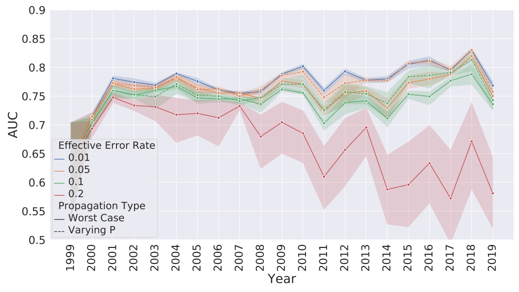


Figure 3: Role of error propagation likelihood p on error amplification strength on MIMIC-IV data for a four layer network. Choosing a smaller p results in less deterioration than expected compared to a threshold that achieves $\gamma' = \gamma p$ while keeping $p = 1$.

year's data while evaluating on next year's data. Further details for the datasets, update procedures, and architectures can be found in Appendix A.

4.1. Classification Threshold Tuning for Long-Term Performance

Deployed models are accompanied by a classification threshold τ which converts a continuous score into a binary 0/1 prediction. τ is typically tuned to maximize a score of interest for a classifier f as follows:

$$\tau = \arg\max_{\tau} s(\mathcal{D}_{\text{val}}, f, \tau)$$

where examples of the scoring function s include F1 Score, Matthew's Correlation Coefficient, or precision and s is chosen to reflect the clinical tradeoff between FPs and FNs. (Tohka and van Gils, 2021). When error amplification is a risk, a model stability tradeoff between FPs and FNs emerges. Namely, the type of error which is capable of flipping labels (we assume it is FPs without loss of generality) poses a challenge to long-term model performance. EA cannot occur if the model's FPR $\gamma = 0$ since by definition there would be no corrupted samples. Choosing a threshold τ that only minimizes γ is trivial, and would result in a model with no clinical value, so the tradeoff between achieving a clinically acceptable value of s while minimizing γ must be considered. This is done by optimizing the long-term performance of the model rather than only the initial performance at deployment. Given a sufficiently large validation set $\mathcal{D}_{\text{val}}$, it can be split into two partitions: $\mathcal{D}_{\text{val}}^{\text{regular}}$ for typical validation set purposes (like early stopping), and $\mathcal{D}_{\text{val}}^{\text{update}}$ for simulating the effect of model updates. $\mathcal{D}_{\text{val}}^{\text{update}}$ is then split according to the model update schedule such that the number of samples in each batch $\mathcal{D}_{\text{val}}^{\text{update}:i}$ is roughly equal to the number of samples expected per batch when gathering data prospectively. We propose the following optimization problem when choosing τ to achieve maximum average long-term performance:

$$\tau = \arg\max_{\tau} \frac{1}{T}\sum_{t=1}^{T} s(\mathcal{D}_{\text{val}}^{\text{update}:t}, f_t, \tau) \tag{1}$$

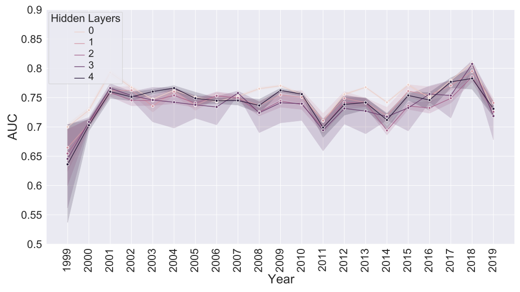
In practice, this is more computationally expensive than the naive threshold selection procedure since each threshold considered requires simulating several updates to obtain long-term performance. To make our proposed threshold selection more tractable, we suggest that experimenters first determine a minimum clinically acceptable score s_{min} for the given task by consulting domain experts and relevant stakeholders. This lower bound on performance can then be used to generate a set of thresholds $\mathcal{T}$ such that the initial model performance for these thresholds is higher than the acceptable lower bound. This gives the following modified version of eq.3:

$$\tau = \arg\max_{\tau \in \mathcal{T}} \frac{1}{T} \sum_{t=1}^{T} s(\mathcal{D}_{\text{val}}^{\text{update}:t}, f_t, \tau) \tag{2}$$

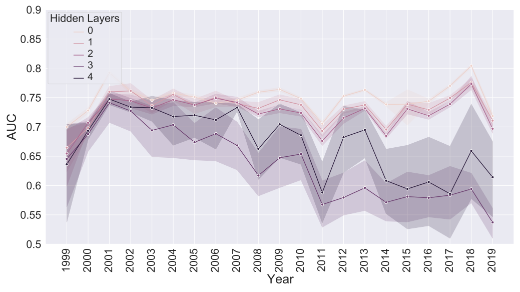
If $|\mathcal{D}_{\text{val}}|$ is too small to simulate updates, then the following heuristic can be used: choose the threshold in $\mathcal{T}$ that results in the smallest FPR γ. This ensures the model will have both clinically acceptable value, and will deteriorate as slowly as possible under this constraint. If the data is highly imbalanced such that the proportion of negative samples is 99%, an FPR of 1% would corrupt 0.99% of the total samples. If the label proportions are inverted such that negative samples are only 1% of the data, then the same FPR results in only 0.01% of samples being corrupted. Thus, a single FPR that limits error amplification across tasks does not exist, but in general a lower FPR is preferred.

To demonstrate the benefit of our proposed threshold selection approach (heuristic version) we first visualize the corresponding FPR that results from maximizing initial F1 score only. We choose F1 score as it is a commonly used combined metric relevant for many classification tasks. This gives us reasonable upper bounds on FPR when conducting the worst-case analysis in the rest of our experiments. Figure 1 reveals that the FPRs corresponding to the highest initial F1 scores are 0.19 for MIMIC-IV, 0.12 for Income Prediction, and 0.36 for Credit Risk. We use 0.3 as the highest FPR in our experiments. Figure 2 compares the long-term model performance when using naive F1 score maximization after each update, and when using choosing a threshold based on our heuristic approach. For both architectures considered, our threshold selection provides better long-term performance on average after the year 2002. This also holds true when evaluating the model in a threshold-dependent manner by considering F1 score in Appendix E.

Lastly, we consider the effect of propagation likelihood p on error amplification. For a given error rate γ, when $p = 0.5$ we intuitively expect long-term model performance to be similar to that of a model which has error rate $\gamma' = \frac{\gamma}{2}$ and $p = 1$. This allows us to investigate non-worst-case settings where humans can correct model errors or subsequent treatment based on model predictions is not capable of always flipping labels. Figure 3 examines the effect of propagation likelihood on error amplification. The *varying-p* models all have the same initial threshold tuned to achieve $\gamma = 0.2$, but p is set to be either [0.5, 0.25, 0.05] resulting in effective FPRs of [0.1, 0.05, 0.01] respectively. The *worst-case* models all use $p = 1$, but have different thresholds tuned to achieve initial FPRs of [0.2, 0.1, 0.05, 0.01]. There is a considerable performance gap between the *varying-p* model which has an *effective* initial FPR of 0.1 and the *worst-case* model which has an actual initial FPR of 0.1. At the time of the first update, the number of errors that propagate is roughly the same between the two models. However, the particular samples whose labels were flipped



(*a*) Initial FPR of $\gamma = 0.1$



(*b*) Initial FPR of $\gamma = 0.2$

Figure 4: Role of model capacity on error amplification strength on MIMIC-IV data for two different initial FPRs. High model capacity has a pronounced effect when γ is larger suggesting that lower capacity models should be chosen in this setting.

do not overlap perfectly between the two models. This is because the model with $\gamma = 0.2$ and $p = 0.5$ corrects half of the model's error's uniformly at random whereas the model with $\gamma = 0.1$ and $p = 1$ reduces the errors specifically around the threshold. Thus, correcting labels post-hoc can be more effective than tuning the threshold to limit the model's error rate, though the two can be combined.

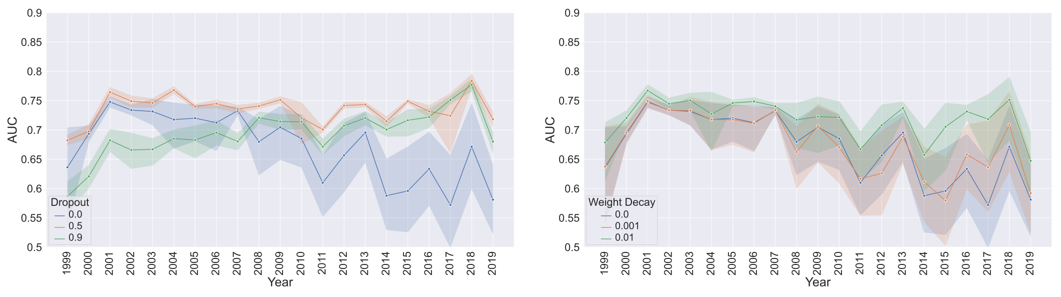
Takeaways

- No single error rate γ which prevents EA for all tasks exists, but a lower γ is preferred.
- ML practitioners should consider the tradeoff between short and long term performance when faced with error amplification by using our proposed threshold selection objective.

4.2. Model Capacity

Another crucial factor which determines the pervasiveness of error amplification is how well a model is able to fit the training data, i.e. model capacity. A hypothesis class $\mathcal{H}$ with high enough capacity as measured by $d_{\text{VC}}(\mathcal{H})$ (VC dimension) will be able to perfectly classify any binary labelling for a given set of samples. This is known as shattering, and implies that hypothesis classes with high d_{VC} are at higher risk of overfitting label noise (Zhang et al., 2016). We explore 5 classification models described in detail in Appendix A.3. To quantitatively compare the neural net architecture capacities, we compute the tight bound of d_{VC} using $\Theta(WU)$ provided by Bartlett et al. (2017) where W is the number of weights and U is the number of neurons. They are all fully connected architectures with increasing depth and width. We use early stopping to prevent overfitting, and no other forms of regularization since that would further confound our analysis.

We visualize the effect of model capacity for two different choices of γ in Fig. 4. For $\gamma = 0.1$ (Fig. 4(a)), the difference in capacity between models does not have a clear effect on performance deterioration since the number of flipped labels relative to the number of clean training samples is small enough. When $\gamma = 0.2$ (Fig. 4(b)), a clearer pattern



(a) Dropout (b) Weight Decay

Figure 5: Effect of regularization on error amplification strength on MIMIC-IV data for a four hidden layer network with an initial FPR of 0.2. Both dropout and weight decay are highly effective at reducing error amplification, and improving initial model performance, with dropout performing slightly better.

emerges. Namely, logistic regression (0 hidden layers, Orange) performs best initially due to its lower sample complexity, and does not deteriorate nearly as much as the deeper models. The deepest 4 layer model (black) outperforms logistic regression when there is no error amplification present once sufficient data is gathered (Appendix Fig. 10), but its high capacity is capable of fitting label noise which leads to much faster model deterioration.

For a more complete picture on how model capacity affects error amplification, regularization has to been considered since it controls the effective number of model parameters. We examine the effect of two common neural network regularization techniques: weight decay and dropout (Krogh et al.; Srivastava et al.). Figure 5 shows that both weight decay and dropout reduce error amplification substantially for a 4 layer deep network whose threshold is tuned to give $\gamma = 0.2$ on MIMIC-IV data. It is important to use a model selection criterion which more heavily penalizes effective model capacity if error amplification is a concern. This is justified by our observations that any small, short-term performance benefit a high capacity model has will disappear after repeated updates. Unfortunately, traditional model selection criteria such as AIC or BIC which more heavily penalize model capacity are not well-suited for overparameterized models such as neural networks (Anders and Korn, 1999). Also, simple counts of model parameters no longer suffice once regularization is used (Moody). We recommend using the following long-term performance criteria similar to eq. 2

$$\mathcal{M} = \underset{\mathcal{M}\in\mathcal{H}}{\arg\max}\, \frac{1}{T}\sum_{t=1}^{T} L(\mathcal{D}_{\text{val}}^{\text{update:}t}, \mathcal{M}_t) \tag{3}$$

which simulates the effect of error amplification using the validation partition, and thus favors models less capable of fitting/amplifying errors induced by the model. L is the data likelihood given the model, and $\mathcal{M}$ is the model.

Overall, our results indicate that constraining model capacity through regularization, and performing model selection on simulated long-term performance can significantly slow error amplification

Takeaways

- Model selection should be done based on simulated long-term performance to avoid risking increased error amplification.
- Standard regularization techniques can significantly reduce error amplification, thus providing benefits beyond improved generalization.

4.3. Model Update Frequency

Intuitively, every model update presents a chance for model parameters to deteriorate which suggests that more frequent small updates may result in worse deterioration than fewer, larger updates. Frequent updating is computationally demanding, but enables the model to adapt better to recent data which is ideal when data shift is present. We partition prospective data into update batches of varying size to see how update frequency impacts error amplification while the total amount of data used to update the model by the final update is the same. We use two different update strategies: data accumulated over time, and a sliding window. We analyze the expected number of corrupted samples for the former strategy, and visualize both strategies in Appendix A.2. Let N_{train} be the number of samples in the original training set, N_{update} be the total number of update samples, i.e. suppose that we would like to evaluate the model's performance after a year's worth of update data gathered. We split N_{update} into a series of update batches such that $N_{\text{update:i}}$ is the number of cumulative update samples when performing update i. For example, if we consider the total number of updates $T = 12$ to represent an update being made each month, then $N_{\text{update:3}}$ represents all update data gathered up to and including month 3. This way, every batch of update data is accumulated and incorporated into the data used to train the model. We assume that each batch of update data contains $\frac{N_{\text{update}}}{T}$ samples for convenience. The expected number of corrupted samples $N_{\text{corr:i}}$ when performing update i is then just

$$\underbrace{\mathbb{E}[N_{\text{corr:i}}]}_{\substack{\text{expected \# of}\\ \text{corrupted samples}}} = \underbrace{\frac{iN^{-}_{\text{update}}}{T(1-\gamma)}}_{\substack{\text{\# neg samples}\\ \text{adjusted for EA}}} - \underbrace{\frac{i}{T}N^{-}_{\text{update}}}_{\substack{\text{observed \#}\\ \text{neg samples}}} = \frac{i\gamma N^{-}_{\text{update}}}{T(1-\gamma)}$$

where N^{-}_{update} is the number of total negatively labeled update samples. We know that there should be more ground truth negatively labeled update samples than observed since error amplification flips them to positive at a rate of γ in this case. Therefore $\frac{iN^{-}_{\text{update}}}{T(1-\gamma)}$ computes how many negatively labeled samples there would be without error amplification, and subtracting $\frac{i}{T}N^{-}_{\text{update}}$ gives the total number of corrupted samples. The expected percentage of corrupted update samples when performing a given update i is

$$\frac{\mathbb{E}[N_{\text{corr:i}}]}{N_{\text{train}} + N_{\text{update:i}}} = \frac{\frac{i\gamma N^{-}_{\text{update}}}{T(1-\gamma)}}{N_{\text{train}} + \frac{i}{T}N_{\text{update}}} = \frac{i\gamma N^{-}_{\text{update}}}{T(1-\gamma)(N_{\text{train}} + \frac{i}{T}N_{\text{update}})}$$

since we use all accumulated data, including the original training set to retrain the model as reflected by $N_{\text{train}} + N_{\text{update:i}}$ in the denominator. To provide a more concrete example, if we choose $N_{\text{update}} = 2N_{\text{train}}$, the percentage of corrupt samples when performing update $\frac{T}{2}$ is:

$$\frac{\gamma N^{-}_{\text{update}}}{(1-\gamma)4N_{\text{train}}} \quad \text{compared to} \quad \frac{\gamma N^{-}_{\text{update}}}{(1-\gamma)3N_{\text{train}}}$$

for the final update. This means that in addition to there being only half the total number of corrupt update samples when performing update $\frac{T}{2}$, the relative weight of those samples is only $\frac{3}{4}$ of what it would be when including all update data at once. We can interpret this as lowering the effective FPR to $\gamma_{\text{effective}} = \frac{1}{2}\gamma$, and making each corrupted sample only $\frac{3}{4}$ as influential on the total loss. Note that even though f_i deviates from f_{orig} with each update, it influences the labels of progressively fewer update samples. The combination of a reduced effective FPR, reduced relative weight per sample, and influence upon progressively fewer samples indicates that the expected snowballing effect of performing multiple updates may not be as severe as initially anticipated when accumulating update data rather than using a sliding window which discards older samples over time.

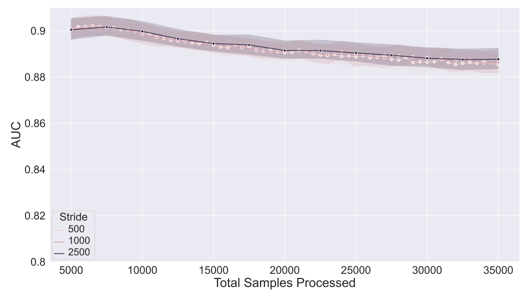
Figure 6 shows how different update frequencies interplay with error amplification on income prediction data, and results on MIMIC-IV are seen in Appendix F. All models in this section use the single hidden layer architecture. If more frequent updates result in more error amplification, we would expect to see the final AUC for the models trained with fewer samples per update (low stride) to be lower than that for models trained with many samples per update (high stride), i.e. the lighter colored lines should be noticeably lower than darker colored lines. This is not the case for the accumulated data update strategy, and two-sided t-tests confirm that there is no significant difference in average performance. However, more frequent updates are much more susceptible to error amplification when using a sliding window strategy. Initially, there is no significant difference as all window strides overlap with the original training data. Once only prospective data is present in the sliding window, retraining is more harmful since flipped samples are a larger percentage of the total samples used to for refitting.

Takeaways

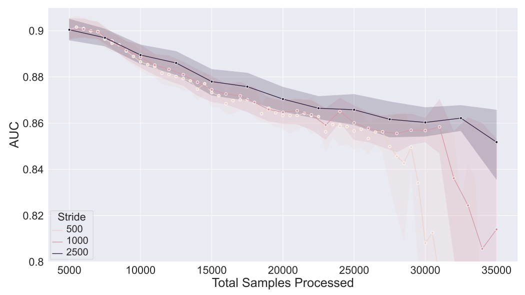
- Given a large enough retrospective dataset that is not discarded as the model is updated, update frequency has a limited effect on error amplification intensity.
- Model updates based on sliding windows which discard data over time are more vulnerable to error amplification when frequent updates are performed.

5. Prevention Methods

We introduce various sample valuation methods and baseline heuristics that have the potential to reduce error amplification. Since error amplification is a relatively new concept, there are no existing methods which have been explicitly designed to deal with it. Nevertheless, the sample valuation methods we consider are intended to simulate a human expert capable of catching and correcting model errors. We focus on ML-based solutions to error amplification, but note that workflow-based solutions also exist. There are some scenarios where it is practical and inexpensive to obtain human expert feedback for each model prediction, for



(*a*) Accumulated data update strategy on Income Prediction data.



(*b*) Sliding window update strategy on Income Prediction data.

Figure 6: Comparison of more frequent smaller updates and less frequent larger updates. No noticeable difference exists between the two extremes (light colored line vs. dark colored line) for the accumulated data update strategy, but error amplification is significantly worse for more frequent updates when using a sliding window strategy. An initial window size of 5000 samples was used.

example when automating diagnostic tests. A clinician can be queried regarding whether they believe that a recommended test was necessary, and this information can be used to correct labels post-hoc. However, in cases where the model drives treatment decisions in such a way that the counterfactual outcome is not observable like in our MIMIC-IV experiments, useful human feedback is difficult to obtain post-hoc. Querying clinicians after a prediction is made is no longer reliable since they may be biased by the prediction. Thus, their medical opinion needs to be recorded prior to the prediction and immediately after to capture the extent of the model's influence. This can only be feasibly done for a fraction of randomly selected patients to avoid workflow disruptions, and measures concordance between clinicians and models to determine prediction reliability. Samples with unreliable predictions can then be excluded when updating the model.

5.1. Heuristics

Dropping Low Confidence Samples An obvious heuristic to use when given a calibrated classifier is to drop low confidence samples. This is closely related to reducing the effective FPR of the model, but is less efficient since the recall of this technique is not perfect, so naturally some correctly labeled samples are dropped as well. If one were to set $\gamma' = \frac{\gamma}{2}$ by either tuning τ, or having access to additional training samples such that classifier performance is improved, then in an update batch comprised of N_- and N_+ samples, only $\frac{\gamma}{2}N_-$ samples would have flipped labels, instead of γN_- samples. We refer to this as *low-conf*.

Dropping All Positively Predicted Samples Not all classifiers are well-calibrated. Calibration is a property that depends on both model class and data used. For example, it is particularly difficult to obtain a calibrated classifier when fitting a highly imbalanced dataset. In such cases, dropping low-confidence samples may not be effective in reducing the percentage of mislabeled update samples, so dropping all positively predicted samples is a more extreme alternative (Adam et al., 2020). This still allows for positively labeled samples as long as the model does not already predict those samples to be positive.

5.2. Sample Valuation

Leave One Out Cross Validation (LOOCV) A simple approach to identifying a potentially noisy sample $\tilde{z}$ part of an update batch $\mathcal{D}_{\text{update}}$ is to update the classifier f both with and without $\tilde{z}$, and observe which version achieves better validation loss:

$$f' = \arg\min_f \mathcal{L}(\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{update}}, f) \qquad f'' = \arg\min_f \mathcal{L}(\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{update}} \setminus \tilde{z}, f)$$

If $\mathcal{L}(\mathcal{D}_{\text{val}}, f') > \mathcal{L}(\mathcal{D}_{\text{val}}, f'')$, then it is likely that $\tilde{z}$ is either an outlier or mislabeled. For this technique to work, $\mathcal{D}_{\text{val}}$ should be balanced since the loss on an imbalanced dataset could be improved by training on a mislabeled sample whose label belongs to the majority class. To identify potentially corrupted update samples, $\mathcal{L}(\mathcal{D}_{\text{val}}, f') - \mathcal{L}(\mathcal{D}_{\text{val}}, f'')$ is computed for each update sample. $\lceil \frac{\gamma N^-_{\text{update}}}{(1-\gamma)} \rceil$ (expected number of corrupted samples) positively labeled samples with the highest difference are excluded from $\mathcal{D}_{\text{update}}$.

Data Shapley Ghorbani and Zou designed a more comprehensive and equitable sample valuation method called Data Shapley to determine if a sample helps or hinders performance. They showed that Data Shapley is more effective than LOOCV at identifying uniform label noise where any sample is equally likely of being randomly mislabeled based on some percentage. Let D be the set of data whose individual samples we are trying to compute a value for (positive values help test performance and negative values hurt test performance), which is simply the training set for our purposes. The value $V(S)$ of a subset $S \subseteq D$ is defined as the test loss for the model learned on S, and is used in the formula

$$\phi_i = \sum_{S \subseteq D - \{i\}} \frac{V(S - \{i\}) - V(S)}{\binom{N-1}{|S|}}$$

where ϕ_i is the value of sample i. This formula considers every possible subset of training data with and without sample i to see if on average including i improves or worsens test performance. We use the more efficient alternative called *G-Shap* to rank potentially mislabeled update samples as the above formulation in infeasible to compute. Let γ be the known false positive rate (FPR) of the model evaluated prior to deployment, and let N^-_{update} be the number of negatively labeled samples in an update batch $\mathcal{D}_{\text{update}}$. Then we compute the *G-Shap* value of all cumulative data, including the update batch $\mathcal{D}_{\text{update}}$ and original training data $\mathcal{D}_{\text{train}}$, and drop $\lceil \frac{\gamma N^-_{\text{update}}}{(1-\gamma)} \rceil$ positively labeled samples with the lowest Data Shapley values from $\mathcal{D}_{\text{update}}$ like in LOOCV.

LRE One can also implicitly reweight samples rather than explicitly trying to identify each noisy sample as way of preventing error amplification. Ren et al. introduced LRE, a second-order optimization technique capable of significantly improving model performance in the presence of uniform label noise. It works by ensuring that training set gradients are geometrically aligned with validation set gradients by dynamically reweighting training samples each update. Given access to a clean validation set, samples corrupted by error amplification would ideally have gradients that do no align with the validation set gradients, would thus be down-weighted by the method. This requires the following update rules

$$\hat{\theta}_{t+1}(\epsilon) = \theta_t - \alpha \nabla_\theta \sum_{i \in \mathcal{T}} \epsilon_i \ell_i(\theta) \Big|_{\theta=\theta_t}$$

$$u_{i,t} = -\eta \frac{\partial}{\partial \epsilon_{i,t}} \frac{1}{|\mathcal{V}|} \sum_{j \in \mathcal{V}} \ell_j(\hat{\theta}_{t+1}(\epsilon)) \Big|_{\epsilon_{i,t}=0} \qquad \tilde{w}_{i,t} = \max(u_{i,t}, 0) \qquad w_{i,t} = \frac{\tilde{w}_{i,t}}{\sum_{j \in \mathcal{T}} \tilde{w}_{j,t}}$$

where $\mathcal{V}$ is the set of validation indices, $\mathcal{T}$ is the set of training indices, θ represents the model parameters (i.e. weights and biases), w is the per-sample weight assigned to each training sample, and $\ell_i(\theta)$ is the loss on a sample with index i evaluated with parameters θ. To make a direct comparison against LOOCV and Data Shapley, we use the per-sample weights at the final epoch of training to determine which update samples hinder validation performance. Specifically, samples with lowest value of w_i, T at the final training iteration T are considered to not improve or reduce validation set performance, and therefore are more likely to be samples whose labels were flipped by error amplification.

5.3. Biased Label Correction

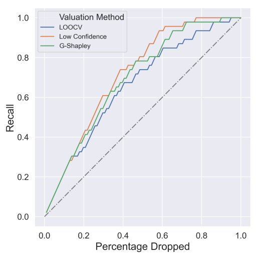
Hausman Rescaling Error amplification can be viewed as a form of systematic measurement error similar to the classic scenario where survey respondents do not understand a question, or are dishonest in their response to sensitive questions such as those about drug use. Hausman et al. propose a way of both estimating the true underlying marginal label rates, and correcting for labeling bias which has the potential to help reduce error amplification for the following reason. We know that error amplification occurs by having the model increase its prediction confidence on noisy samples. For example, a negative sample x is corrupted when the model's prediction confidence $f(x)$ is greater than the prediction threshold τ. Say that $\tau = 0.5$, and $f(x) = 0.51$. The noisy label is $\hat{y} = 1$, so when trained on this noisy sample, the model is encouraged to increase its prediction confidence and predict $f(x) = 1$. Hausman et al. shift the probability predicted by a classifier to take into account a known underlying error rate, in our case the FPR γ, as to better approximate the true $p(y)$ using the noisy version $\hat{p}(y)$ via

$$\alpha_0 = p(\hat{y} = 1 | y = 0) \qquad \alpha_1 = p(\hat{y} = 0 | y = 1)$$
$$f(x) = \alpha_0 + (1 - \alpha_0 - \alpha_1) f(x)$$
$$= (1 - \alpha_0) f(x) + \alpha_0$$

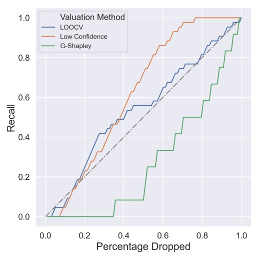
where α_0 is the probability of a negative sample being flipped positive (FPR in our case), α_1 is the probability of a positive sample being flipped negative (0 in our case). Practically speaking, this discourages the model from making very high confidence predictions for positively labeled samples. It does so by artificially increasing the probability predicted by the model for positively labeled samples thereby reducing the binary cross entropy loss.

6. Regularization Effectiveness

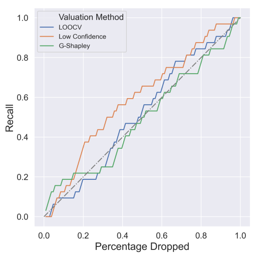
For the methods that explicitly attempt to identify noisy samples, such as *low-conf*, *G-Shap* , *loocv*, we evaluate their recall as a function of what percentage of positively labeled update samples are dropped. Details can be found in Appendix B. Figure 7 shows the recall of all three methods on the income and credit risk prediction datasets. The size and temporal nature of MIMIC-IV makes it computationally infeasible to use for this experiment, so we exclude it. Uniform label noise in the income prediction data is identified with better than random performance by all three methods as expected. Error amplification noise is more difficult to detect, and *low-conf* is the best method. Credit risk data is more difficult to rank for both uniform and error amplification noise which is to be expected since the task is more difficult with a test AUC 0.75 vs. 0.90 for income prediction. *low-conf* has the highest recall for uniform noise, with *loocv* and *G-Shap* now only does as well as random chance. None of the methods can identify error amplification noise better than random chance on this data.



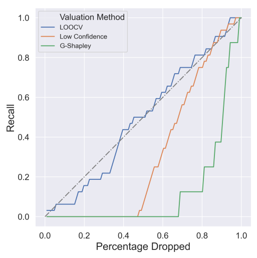
(*a*) Income Prediction data with uniform noise.



(*b*) Income Prediction data with error amplification noise.



(*c*) Credit Risk data with uniform noise.



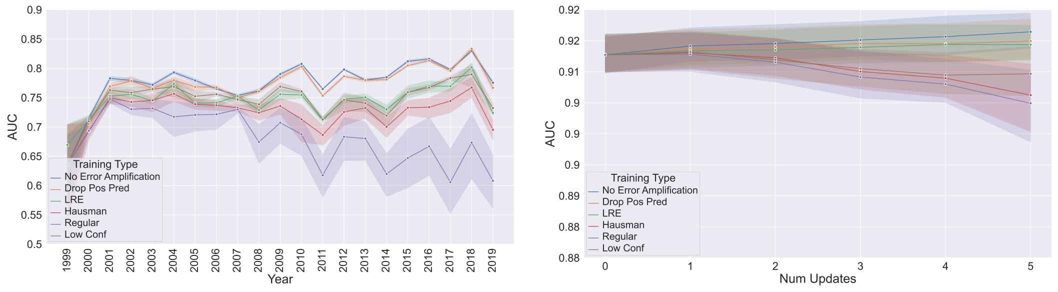
(*d*) Credit Risk data with error amplification noise.

Figure 7: Recall of sample valuation methods for identifying uniform label noise, and label noise caused by error amplification. Error amplification noise is the most difficult to identify for both datasets, and none of the methods do well for it.

Based on the recall results, we exclude *G-Shap* and *loocv* since their effectiveness relative to their computational requirements is low. On MIMIC-IV, it is clear that dropping positively predicted samples baseline is the most effective according to Figure 8 (a). LRE is also a good option in reducing feedback loop intensity as well, but a caveat not shown in Figure 8 (a) is that when error amplification is not present, LRE does worse than no regularization since it slows learning on new data and keeps the model more static over time. The next most effective method is Hausman regularization. This is promising because it is computationally inexpensive, and can be easily implemented. Thus, we recommend training with Hausman regularization if error amplification is a risk. A similar ranking exists on the Income Prediction dataset (Figure 8 (b)), though Hausman regularization is not as effective here.

7. Discussion

Deploying machine learning in practice remains a challenging process due to data shift and potential regression in performance that may occur with each model update Xie et al. (2021). One reason for performance regression is error amplification, especially when using a high capacity model that can easily overfit noisy samples causing worse generalization over time.



(*a*) MIMIC-IV data for a 4 hidden layer MLP, $\gamma = 0.2$

(*b*) Income Prediction data 4 hidden layer MLP, $\gamma = 0.2$

Figure 8: Comparison of regularization techniques used to reduce error amplification. Hausman adjustment improves stability on both datasets, requires minimal overhead, and can be combined with the other methods. Dropping positively predicted samples performs best, followed by LRE which is the most computationally expensive. Dropping low confidence positive also helps stabilize model performance.

Measuring performance deterioration is not straightforward due to the model error creep into the labels which can falsely indicate that the model is improving over time. The demand for continuously validating the performance of ML models once deployed has lead to the development of tools for monitoring data quality and ensuring prediction stability (Caveness et al., 2020). But none of these tools can address error amplification effectively. Developing strategies for dealing with error amplification is a crucial requirement for safe and effective deployment of ML models in the real world, especially in high risk settings.

Many previous approaches addressing deployment of ML models relied on simplifying assumptions. Solving error amplification in the general case requires approaches that make minimal assumptions, acknowledging that both $P(y)$ and $P(x|y)$ are changing over time. While this is a very difficult problem to solve, our work provides optimistic observations regarding how this phenomenon can be slowed down by making careful design choices prior to deployment. Thresholds can be tuned in such a way that model utility is only slightly reduced while error amplification strength is more than halved. Additionally, model capacity can be constrained to favor higher bias, lower variance models as we have demonstrated that such models are much less impacted by error amplification noise. A limitation of our work is that we aim to provide general recommendations, so we do not make strong modelling or data assumptions necessary for robust theoretical results. Therefore, it is possible that the observations made will not hold for all model/dataset combinations. In settings where data can be reviewed by experts, and labels can be corrected at a reasonable cost, error amplification can be detected faster and can be reduced by correcting noisy labels with post-hoc human predictions. Our research demonstrates that there is no free lunch when it comes to addressing error amplification. However, having models with a low enough error rate prevents a critical percentage of update data from being corrupted, which in turn ensures that error amplification will not cause model deterioration over time.

References

George Alexandru Adam, Chun-Hao Kingsley Chang, Benjamin Haibe-Kains, and Anna Goldenberg. Hidden Risks of Machine Learning Applied to Healthcare: Unintended Feedback Loops Between Models and Future Data Causing Model Degradation. Technical report, sep 2020. URL http://proceedings.mlr.press/v126/adam20a.html.

Ulrich Anders and Olaf Korn. Model selection in neural networks. *Neural Netw.*, 12(2): 309–323, March 1999.

Peter L Bartlett, Nick Harvey, Christopher Liaw, Abbas Mehrabian, and Mehrabian Bartlett. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks *. 2017. URL http://proceedings.mlr.press/v65/harvey17a.html;.

Emily Caveness, Paul Suganthan, Zhuo Peng, Neoklis Polyzotis, Sudip Roy, and Martin Zinkevich. TensorFlow data validation: Data analysis and validation in continuous ML pipelines. *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, page 4, 2020.

Danielle Ensign, Sorelle A Friedler, Scott Neville, Carlos Scheidegger, Suresh Venkatasubramanian, and Christo Wilson. Runaway Feedback Loops in Predictive Policing *. Technical report, 2018. URL https://github.com/algofairness/.

Amirata Ghorbani and James Zou. Data Shapley: Equitable Valuation of Data for Machine Learning. Technical report.

J. A. Hausman, Jason Abrevaya, and F. M. Scott-Morton. Misclassification of the dependent variable in a discrete-response setting. *Journal of Econometrics*, 87(2):239–269, dec 1998. ISSN 0304-4076. doi: 10.1016/S0304-4076(98)00015-3.

Anton S Khritankov. ANALYSIS OF HIDDEN FEEDBACK LOOPS IN CONTINUOUS MACHINE LEARNING SYSTEMS A PREPRINT 1. doi: 10.1007/978-3-030-65854-0_5. URL https://doi.org/10.1007/978-3-030-65854-0_5.

Anders Krogh, John A Hertz, Nordita Blegdamsvej, and Dk-2100 Copenhagen. A simple weight decay can improve generalization. https://proceedings.neurips.cc/paper/1991/file/8eefcfdf5990e441f0fb6f3fad709e21-Paper.pdf. Accessed: 2022-7-21.

James Liley, Samuel Emerson, Bilal Mateen, Catalina Vallejos, Louis Aslett, and Sebastian Vollmer. Model updating after interventions paradoxically introduces bias. 130:3916–3924, 2021.

Zachary C Lipton, Yu-Xiang Wang, and Alexander J Smola. Detecting and Correcting for Label Shift with Black Box Predictors. Technical report, 2018.

Nikhil Malik. Does Machine Learning Amplify Pricing Errors in Housing Market? : Economics of ML Feedback Loops, sep 2020. URL https://papers.ssrn.com/abstract=3694922.

Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. Feedback Loop and Bias Amplification in Recommender Systems. In

International Conference on Information and Knowledge Management, Proceedings, pages 2145–2148, New York, NY, USA, oct 2020. Association for Computing Machinery. ISBN 9781450368599. doi: 10.1145/3340531.3412152. URL https://dl.acm.org/doi/10.1145/3340531.3412152.

John E Moody. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. https://proceedings.neurips.cc/paper/1991/file/d64a340bcb633f536d56e51874281454-Paper.pdf. Accessed: 2022-6-16.

Bret Nestor, Matthew B A Mcdermott, Willie Boag, Gabriela Berner, Tristan Naumann, Michael C Hughes, Anna Goldenberg, and Marzyeh Ghassemi. Feature Robustness in Non-stationary Health Records: Caveats to Deployable Model Performance in Common Clinical Machine Learning Tasks. 2019. URL https://mimic.physionet.org/mimicdata/carevue/.

Bret Nestor, Liam G McCoy, Amol Verma, Chloe Pou-Prom, Joshua Murray, Sebnem Kuzulugil, David Dai, Muhammad Mamdani, Anna Goldenberg, and Marzyeh Ghassemi. Preparing a clinical support model for silent mode in general internal medicine. In Finale Doshi-Velez, Jim Fackler, Ken Jung, David Kale, Rajesh Ranganath, Byron Wallace, and Jenna Wiens, editors, *Proceedings of the 5th Machine Learning for Healthcare Conference*, volume 126 of *Proceedings of Machine Learning Research*, pages 950–972. PMLR, 2020.

Juan C Perdomo, Tijana Zrnic, Celestine Mendler-Dünner, and Moritz Hardt. Performative Prediction. Technical report, 2020.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to Reweight Examples for Robust Deep Learning. *35th International Conference on Machine Learning, ICML 2018*, 10:6900–6909, mar 2018. URL http://arxiv.org/abs/1803.09050.

M Saerens, P Latinne, and C Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. *Neural computation*, 14(1):21–41, jan 2002. ISSN 0899-7667. doi: 10.1162/089976602753284446. URL https://pubmed.ncbi.nlm.nih.gov/11747533/.

D Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. Hidden Technical Debt in Machine Learning Systems. Technical report.

Ayan Sinha, David F Gleich, and Karthik Ramani. Deconvolving Feedback Loops in Recommender Systems. Technical report.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. https://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf. Accessed: 2022-7-21.

Jussi Tohka and Mark van Gils. Evaluation of machine learning algorithms for health and wellness applications: A tutorial. *Comput. Biol. Med.*, 132:104324, May 2021.

Yuqing Xie, Yi-An Lai, Yuanjun Xiong, Yi Zhang, and Stefano Soatto. Regression bugs are in your model! measuring, reducing and analyzing regressions in NLP model updates. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Stroudsburg, PA, USA, 2021. Association for Computational Linguistics.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, nov 2016. URL https://arxiv.org/abs/1611.03530v2.

Appendix A. Experimental Setup

A.1. Datasets

A.1.1. Temporal

MIMIC-IV is a clinical dataset gathered over two decades from an ICU in Boston: https://physionet.org/content/mimiciv/0.4/. The task is mortality prediction over the next 48 hours using clinical and demographic variables 31k samples and 48 features. We follow the same training procedure as (Adam et al., 2020) where the years 1996-1998 are used as the retrospective data $\mathcal{D}_{\text{train}}$, and the remaining data is considered prospective and is used to evaluate and update the model on a yearly basis. Further details can be found in their work. When analyzing the effect of update frequency, we shuffle the data and treat it like a non-temporal dataset so that we can control the number of samples at each update since otherwise this varies year to year. In this case, we have held-out test set rather than doing prequential evaluation where performance is evaluated on next year's patients.

A.1.2. Non-Temporal

The Income Prediction dataset is a benchmark to predict whether an individual is likely to earn over or under \$50k a year based on occupation, education, and marital status. There are 49k samples with 14 features: https://archive.ics.uci.edu/ml/datasets/adult.

The last dataset is a Credit Risk (bad/good) classification dataset using features such as past credit history, purpose for loan, and so on. The dataset consists of 1000 samples and 20 features: https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data). We use unconventional data splits of 20/20/40/20 for training/validation/updating/testing respectively on the non-temporal datasets. This is done to ensure model performance improves when there is no error amplification, and worsens when there is. Since both tasks are relatively easy, using just 20% of the training data gives acceptable performance. Having sufficient update data is crucial for error amplification to occur, hence the large update percentage of 40%.

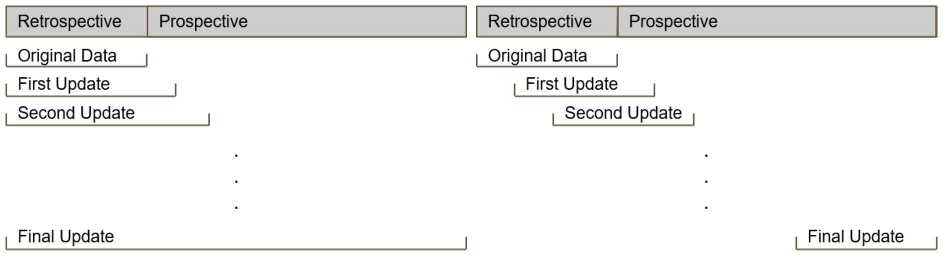
A.2. Retraining Procedure

We use two retraining approaches throughout seen in figure 9, both of which involve refitting the model from scratch which helps avoid having to define separate hyperparameters for learning on retrospective and prospective data. The `accumulating data` strategy uses all data gathered up to timestep t to obtained new parameters

$$\theta_t = \arg\min_{\theta} \mathop{\mathbb{E}}_{(x,\hat{y})\sim\mathcal{D}_{\text{accumulated:t}}} L(\hat{y}, f_\theta(x))$$

$$\mathcal{D}_{\text{accumulated:t}} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{update:0}} \cup \ldots \cup \mathcal{D}_{\text{update:t}-1}$$

where $\mathcal{D}_{\text{train}}$ is the original retrospective dataset, and $\mathcal{D}_{\text{update:i}}$ represents prospective data gathered at update i. The `sliding window` strategy instead only uses data from the current window which initially has some overlap with the training data, but eventually



(a) Accumulating Data Strategy (b) Sliding Window Strategy

Figure 9: The update strategies we use throughout our experiments.

contains only prospective data

$$\theta_t = \arg\min_{\theta} \mathop{\mathbb{E}}_{(x,\hat{y})\sim\mathcal{D}_{\text{sliding:t}}} L(\hat{y}, f_\theta(x))$$

.

Furthermore, each update the model threshold τ is refit either using a greedy score maximizing approach, or our proposed approach.

A.3. Architectures

We use 5 different fully connected architectures throughout our experiments, each of which have ReLU as the hidden activation function. The networks are all optimized using Adam with a learning rate of 0.001, and an early stopping threshold of 50 epochs. 10 random seeds are used for each model to estimate uncertainty information, and reveal stability. Since the architectures and datasets are relatively small, optimization is done full-batch rather than using mini-batches. We use the tight bound $\Theta(WU)$ on d_{VC}.

0 Hidden Layers - $\mathcal{A}_1$ A logistic regression model with no hidden layers. $d_{\text{vc}}(\mathcal{A}_1) = D+1$ where D is the number of input dimensions.

1 Hidden Layer - $\mathcal{A}_2$ A single hidden layer deep network $H_1 = 10$ hidden units. $W = D \times 10 + 10$, $U = 10$, so
$c_1 \cdot 100(D+1) \leq d_{\text{VC}}(\mathcal{A}_2) \leq c_2 \cdot 100(D+1)$ where c_1, c_2 are positive constants, and D is the number of input dimensions.

2 Hidden Layers - $\mathcal{A}_3$ A two hidden layer deep network $H_1 = 20$, $H_2 = 10$ hidden units. $W = D \times 20 + 20 \times 10 + 10$, $U = 20 + 10$, so
$c_1 \cdot 300(2D+21) \leq d_{\text{VC}}(\mathcal{A}_3) \leq c_2 \cdot 300(2D+21)$ where c_1, c_2 are positive constants, and D is the number of input dimensions.

3 Hidden Layers - $\mathcal{A}_4$ A three hidden layer deep network $H_1 = 400$, $H_2 = 200$, $H_3 = 100$ hidden units. $W = D \times 400 + 400 \times 200 + 200 \times 100 + 10$, $U = 400 + 200 + 100$ so
$c_1 \cdot 7000(400D + 10001) \leq d_{\text{VC}}(\mathcal{A}_4) \leq c_2 \cdot 7000(400D + 10001)$ where c_1, c_2 are positive constants, and D is the number of input dimensions.

4 Hidden Layers - $\mathcal{A}_5$ A four hidden layer deep network $H_1 = 800$, $H_2 = 400$, $H_3 = 200$, $H_4 = 100$ hidden units. $W = D \times 800 + 800 \times 400 + 400 \times 200 + 200 \times 100 + 10$, $U = 800 + 400 + 200 + 100$ so
$c_1 \cdot 1500(800D + 420010) \leq d_{\mathrm{VC}}(\mathcal{A}_5) \leq c_2 \cdot 1500(800D + 420010)$ where c_1, c_2 are positive constants, and D is the number of input dimensions.

A.4. Code Availability

The code repository can be found here.

Appendix B. Sample Valuation Recall Details

For *g-shap* and *loocv*, we use the validation set to measure the performance difference with and without including a particular sample. Logistic regression is used for the recall experiments in this section because it tends to be well-calibrated without post-hoc calibration which is a requirement for *low-conf*.

Since *g-shap* and *loocv* are computationally intensive compared to regular training, a subset of 200 update samples is used to evaluate their effectiveness. Uniform label noise is used as a sanity check to evaluate the effectiveness of the methods since it is one of the easiest forms of noise to identify. We generate such noise in an asymmetric manner by flipping negative samples to positive for a random subset of negative samples in the update data.

Appendix C. Model Capacity

C.1. Benefit of High Capacity Models

Figure 10 shows that when error amplification is present, the higher capacity 4 hidden layer model is much more effective than logistic regression (0 hidden layers) once sufficient data has been accumulated starting in 2002.

Reproducibility Details: All the experiments are done on MIMIC-IV, training on samples from 1996-1998 and evaluating/updating prequentially thereafter on a yearly basis.

All architectures were fit using the same optimization hyperparameters: Adam optimizer with a learning rate of 0.001, early stopping after 50 iterations, and a maximum of 10,000 full batch learning.

For the Dropout experiments, Dropout was added after every hidden layer. We searched over a range of Dropout probabilities $p \in [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$ and found 0.5 to be the most effective.

The weight decay experiments used a range of $\lambda \in [1e{-}6, 1e{-}5, 1e{-}4, 1e{-}3, 1e{-}21e{-}1, 1]$, with 1e-2 being the most effective.

Appendix D. Update Frequency

Reproducibility For the experiments analyzing the effect of update frequency, we shuffle the data for MIMIC-IV since the number of samples per year is not consistent, and the size of each update must be controlled. We use a data split of 20/10/60/10 for training/validation/updating/testing on all datasets in this section. An initial window size of 5000

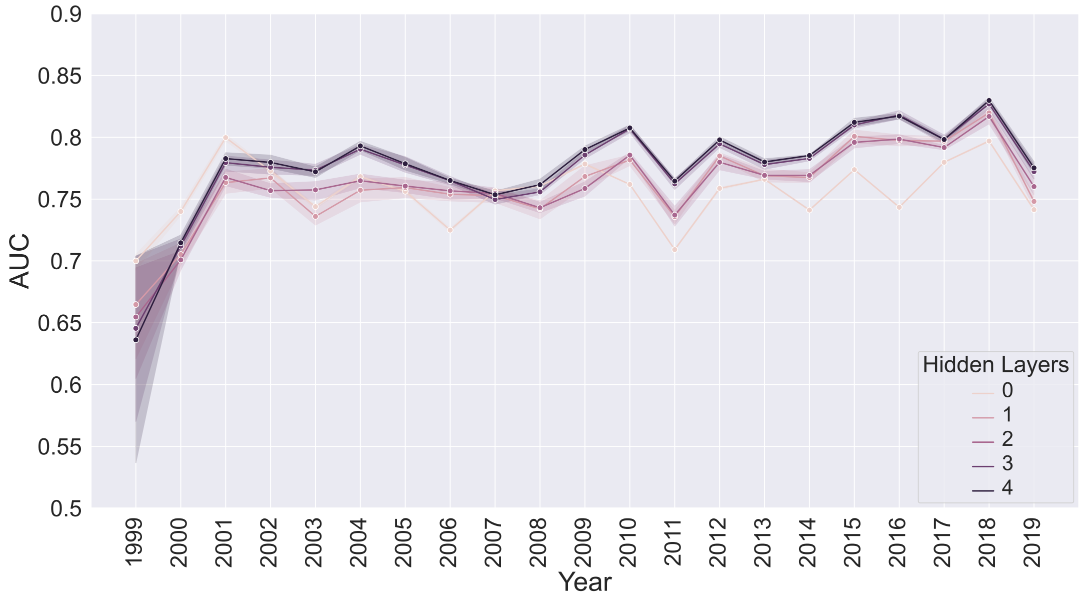


Figure 10: Effectiveness of higher capacity models on MIMIC-IV data when error amplification is not present. It is clear that higher capacity models are preferred once sufficient data is gathered unlike when error amplification is present.

samples was used for both the accumulating window, and sliding window update strategies. Optimization hyperparameters are the same as all other sections.

Appendix E. Short vs. Long Term Threshold Optimization

Our proposed threshold selection procedure reduces error amplification by choosing a threshold which limits the propagating error type while still achieving clinical utility. We examine if the benefit of this approach still exists when evaluating the model in a threshold-dependent manner. Fig 11 shows that greedy F1 score maximization initially outperforms our approach as expected. However, over time its higher FPR leads to more errors propagating, and more degradation with each update such that even the heuristic version of our approach ends with a better F1 score even though it does not explicitly maximize F1.

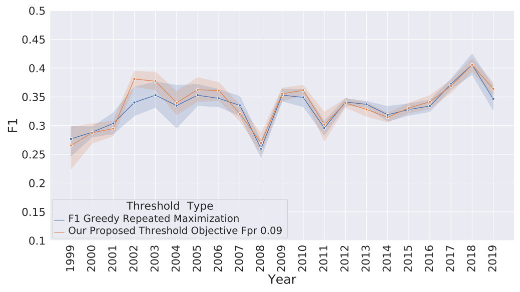
Reproducibility Details: For the F1 vs. FPR figures, we use the following data splits

1. MIMIC-IV: Train on all samples from 1996-1998, evaluate thresholds on data from 1999.

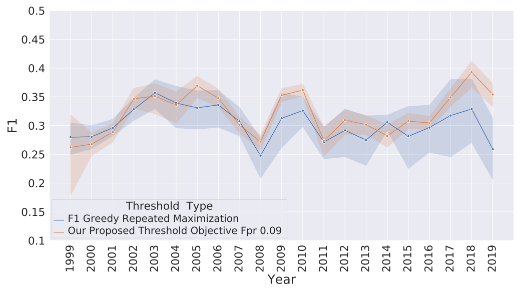
2. Income Prediction, Credit Risk: Train on a randomly selected subset using 20% of all samples, evaluate thresholds on another randomly selected subset using 20% of all samples.

Logistic regression models were fit to the data for these figures using the Adam optimizer with a learning rate of 0.001, early stopping after 50 iterations, and a maximum of 10,000 full batch learning iterations in PyTorch.

The figures comparing our proposed threshold approach against greedily maximizing F1 score at each step use the data split details from Appendix A. When updates are performed, thresholds are refit using 20% of all data gathered so far. Importantly, our threshold selection approach, which on MIMIC-IV used a threshold achieving an initial FPR of 0.09, tries to maintain that same FPR across all updates.



(a) Single hidden layer

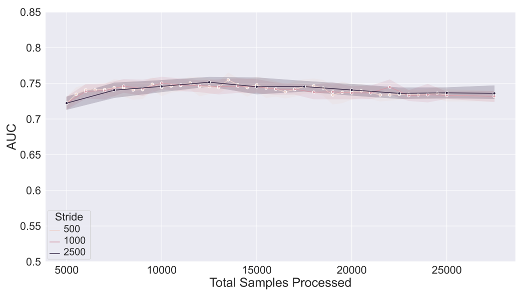


(b) Four hidden layers

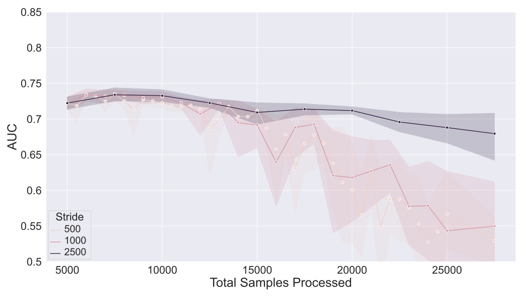
Figure 11: Threshold-dependent evaluation of greedy F1 score maximization and our long-term performance threshold optimization objective. For the deeper model that is more susceptible to error amplification, our approach achieves a better long-term F1 score than naively maximizing F1 score at each update.

Appendix F. Update Frequency

The difference between the accumulated data and sliding window strategies also exists on MIMIC-IV data (Fig 12). There is no significant performance difference between frequent, small updates (low stride) and infrequent, large updates (high stride) when using the accumulated data update strategy. However, a clear pattern emerges for the sliding window strategy where less frequent updates result in less model degradation over time. The reason why performance seems stable for the accumulated data strategy compared to other experiments in the paper done on MIMIC-IV is because we have shuffled the data, and created a held-out set rather than doing a temporal evaluation in a prequential manner. See Appendix A.2 for details. If we repeat this experiment without simulating error amplification, there is still no significant performance difference between update frequencies for either update strategy, and also there is a clear improvement in performance over time as more samples are gathered for the accumulated update strategy (Fig 13)

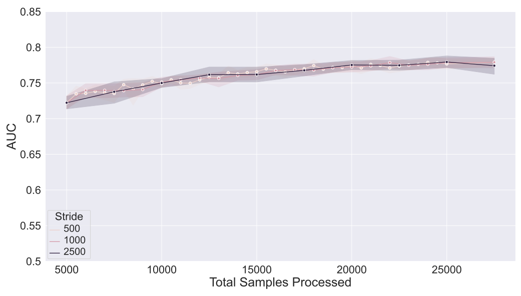


(*a*) Accumulated data update strategy on MIMIC-IV data.

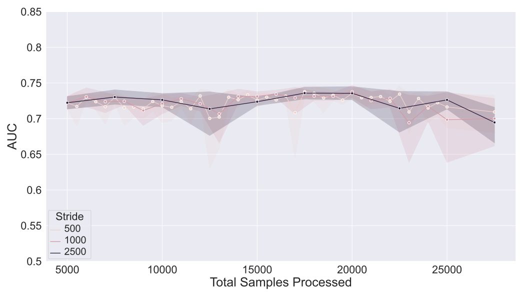


(*b*) Sliding window update strategy on MIMIC-IV data.

Figure 12: Comparison of more frequent smaller updates and less frequent larger updates on MIMIC-IV data when error amplification is present. No noticeable difference exists between the two extremes (light colored line vs. dark colored line) for the accumulated data update strategy, but error amplification is significantly worse for more frequent updates when using a sliding window strategy. An initial window size of 5000 samples was used.



(*a*) Accumulated data update strategy on MIMIC-IV data.



(*b*) Sliding window update strategy on MIMIC-IV data.

Figure 13: Comparison of more frequent smaller updates and less frequent larger updates on MIMIC-IV data when there is no error amplification. No noticeable difference exists between the two extremes (light colored line vs. dark colored line) for either update strategy. An improvement in performance over time is seen as expected for the accumulated data strategy due to the larger sample size and no error propagation. An initial window size of 5000 samples was used.