(Don't) Forget About It: Forgetting-Penalized Supervised Learning*

Gaurav Sood†

July 16, 2025

Abstract

During continued training, a machine-learning model can begin misclassifying examples it previously labeled correctly—a within-task instability often called model regression. While catastrophic forgetting across different tasks has been studied for decades, standard supervised objectives still optimize only average accuracy and ignore these per-example reversals, even though each flip can trigger costly downstream work in production pipelines that rely on stable behavior. We propose two lightweight, differentiable regularizers that embed a "do-no-harm" bias directly into the loss. The flip penalty adds cost whenever an example that was correct with a safety margin in the previous epoch becomes incorrect, and the soft Pareto penalty more gently discourages any increase in per-example loss. Both methods track just one bit of historical state per sample and require no extra passes over the data, so they drop seamlessly into existing cross-entropy training loops. In a proof-of-concept study on the Adult-Income benchmark, enabling either penalty after a 10-epoch warm-up cut forgetting events on a fixed evaluation set by 95–98% compared with an unregularized baseline. The soft Pareto variant kept test accuracy within 0.5% of the baseline, showing that stability and performance need not trade off sharply. Because the penalties are simple to implement and add negligible computation, they offer a practical safeguard for production systems—especially human-in-the-loop settings—where consistency on known-good cases is valued alongside headline accuracy.

1 Introduction

Machine-learning models deployed in production are routinely retrained or fine-tuned as new data arrive. Although these updates generally *improve* overall accuracy, they can sometimes

*https://github.com/finite-sample/pareto-gd.

†Gaurav can be reached at gsood07@gmail.com

cause the same model to misclassify examples it previously labeled correctly. We refer to each such reversal as *model regression*. This phenomenon resembles catastrophic forgetting (French, 1999) but occurs *within a single supervised task*, rather than across clearly separated tasks in continual-learning benchmarks. While catastrophic forgetting has inspired a rich body of work in multi-task settings (Kirkpatrick et al., 2017; Zenke et al., 2017), the practical consequences of model regression in production pipelines have received far less attention.

Model regression is more than a statistical curiosity: in human-in-the-loop systems it incurs operational costs. When performance drops for a specific subgroup, organizations must reallocate domain specialists, launch retraining programs, or absorb higher error-handling expenses. Section 2 quantifies these costs through a staffing-optimization case study and shows that even modest regressions can erase the financial gains of an accuracy update.

Standard supervised learning objectives minimize average loss and provide no incentive to preserve correctness on *known-good* examples. Consequently, training dynamics may sacrifice stability for small aggregate improvements, exposing production systems to disruptive swings in behavior.

In this paper, we propose two lightweight regularizers that address this gap:

1. **Flip Penalty**—adds a differentiable cost whenever an example that was correct with a safety margin in the previous epoch becomes incorrect.

2. **Soft Pareto Penalty**—penalizes any increase in per-example loss, encouraging monotonic improvement without freezing progress on harder samples.

Both methods require only a single bit of historical state per training example and no additional passes over the data, making them easy to integrate into existing cross-entropy pipelines.

Our contributions are threefold:

1. We formalize *model regression* and provide an economic framework that links per-example reversals to real operational costs.

2. We introduce two differentiable, example-level penalties that substantially reduce regression with negligible computational overhead.

3. We empirically demonstrate the effectiveness of these penalties on the Adult-Income benchmark and discuss their applicability to production systems.

2 Operational Cost Case Study: Customer-Operations Staffing

To ground the economic stakes of model regression, consider a financial services firm that uses an ML classifier to triage incoming customer requests and detect cases requiring human review. Requests are routed into categories—*account inquiries*, *fraud alerts*, *loan applications*, *technical support*, and so on. Each category has a dedicated team of specialists who intervene when the model's confidence is low or when it commits an error.

2.1 Economic Drivers

- **Specialization efficiency**: Employees work fastest and most accurately within their domain expertise (e.g., fraud analysts versus loan officers).
- **Retraining cost**: Moving workers between domains requires expensive retraining that temporarily reduces productivity.
- **Hiring and separation cost**: Adjusting headcount entails recruiting, onboarding, or severance expenses.
- **Accuracy benefit**: Higher model accuracy lowers the total volume of human interventions, reducing labor demand across all categories.

2.2 The Regression Cost Problem

When updating the ML model from version M_A to M_B, the company faces a trade-off. While M_B may achieve higher overall accuracy than M_A, it may perform worse on specific subgroups or categories. This creates two types of costs:

Staffing Realignment Costs: If model M_B has worse performance on fraud detection but better performance on loan processing, staff must be moved from loan processing to fraud detection roles, incurring retraining costs.

Accuracy Benefits: If model M_B has sufficiently higher overall accuracy, fewer total staff are needed, creating cost savings.

Let $C_{move}(M_A, M_B)$ denote the cost of staff realignment when transitioning from M_A to M_B, and let $C_{staff}(M)$ denote the ongoing staffing cost for model M. The company should only deploy M_B if:

$$C_{staff}(M_A) - C_{staff}(M_B) > C_{move}(M_A, M_B) \tag{1}$$

2.3 Forgetting-Penalized Training as Economic Optimization

This business case provides clear economic justification for forgetting-penalized training. Rather than accepting any model with higher overall accuracy, the company can use penalties to constrain model updates such that:

1. **Subgroup Performance Preservation**: Penalize decreases in performance on individual categories to minimize staff reallocation needs
2. **Minimum Improvement Threshold**: Only accept model updates if the overall accuracy improvement exceeds a threshold τ that justifies operational disruption
3. **Pareto-Aware Updates**: Prefer model improvements that help some categories without harming others

The penalty-based objective becomes:

$$\mathcal{L}_{economic}(\theta) = \mathcal{L}_{std}(\theta) + \lambda_{econ} \sum_{k=1}^{K} \max(0, \ell_k(M_A) - \ell_k(M_\theta)) \tag{2}$$

where $\ell_k(M)$ represents the loss of model M on category k, and the penalty term discourages performance decreases on any category.

2.4 Broader Implications

This operational perspective suggests that "regression" in machine learning should be understood not merely as a technical phenomenon, but as an economic externality. The cost of operational disruption from model updates may outweigh the benefits of marginal accuracy improvements, particularly in:

- **Human-in-the-loop systems** where model changes affect workforce allocation
- **Multi-stakeholder environments** where different groups depend on model performance for specific use cases
- **Regulated industries** where model changes require compliance reviews and approval processes
- **Customer-facing applications** where inconsistent model behavior affects user experience and trust

This economic framing provides a principled foundation for determining when forgetting prevention is worthwhile and how to set appropriate penalty weights λ based on operational costs rather than purely technical considerations.

3 Related Work

3.1 Example Forgetting as a Dataset Diagnostic

Toneva et al. (2019) introduced the notion of a *forgetting event*—a correct $\rightarrow$ incorrect transition for a single example during SGD—and showed that (i) forgetting counts follow a heavy-tailed distribution, (ii) the ranking of (un)forgettable examples is architecture-invariant, and (iii) pruning the *never-forgotten* fraction of CIFAR-10 does not harm test accuracy. Follow-up work leverages forgetting scores for data curation (Paul et al., 2021) and long-tailed re-weighting (Yu and Chen, 2022). In contrast, we embed a *differentiable* forgetting cost directly into the loss to prevent harmful flips in the first place.

3.2 Catastrophic forgetting and continual learning

Catastrophic forgetting—loss of performance on earlier tasks while learning new ones—was first noted by French (1999). Continual-learning research has since produced three main families of defenses:

Parameter-level regularizers. Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), Memory-Aware Synapses (MAS) (Zenke et al., 2017), and Synaptic Intelligence (SI) (Zenke et al., 2017) estimate parameter importance and penalize updates on important weights. EWC, for example, uses the diagonal Fisher information,

$$F_i \;=\; \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{old}}}\Big[\big(\tfrac{\partial}{\partial\theta_i}\log p(y \mid x,\theta)\big)^2\Big],$$

and adds a quadratic penalty $\frac{\lambda}{2}\sum_i F_i(\theta_i - \theta_i^\star)^2$ to the new task loss. These methods are effective when task boundaries are known, but require storing (or recomputing) importance estimates and do not directly address within-task regression.

Rehearsal buffers. Methods such as Gradient Episodic Memory (Lopez-Paz and Ranzato, 2017) replay a subset of past examples while learning new ones, thereby preserving previous performance at the cost of extra memory and computation.

Architectural expansion. Progressive networks (Rusu et al., 2016) and related approaches allocate new capacity for each task, sidestepping interference but sacrificing parameter efficiency.

Our work differs in scope: we remain within *one* supervised task and operate at the *output level*, tracking only one bit per example and requiring no task boundaries or replay.

3.3 Multi-objective and Pareto-optimal training

Multi-task learning can be cast as a multi-objective problem in which each task loss is an independent objective. Lin et al. (2019) solve a set of constrained sub-problems in parallel to approximate the Pareto front, producing a spectrum of accuracy–fairness (or other) trade-offs that users can select post-training. Navon et al. (2021) go further, training a hypernetwork that maps a preference vector to weights, thus generating the entire Pareto front in a single run.

These methods aim to *offer choices* along a frontier of trade-offs. In contrast, our penalties pursue a single, conservative update: they bias SGD toward changes that improve—or at least do not worsen—every example's loss. The goal is operational stability, not exploration of a full trade-off surface, making our approach better suited to production pipelines that value consistency on known-good cases.

3.4 Summary

Prior work either analyzes example-level forgetting after training, protects entire tasks via parameter or rehearsal methods, or explores Pareto trade-offs across tasks.

Our contribution is orthogonal: a lightweight, example-level regularizer that prevents within-task regression *during* training and aligns directly with operational cost constraints in real-world deployments.

4 Methods

4.1 Problem Formulation

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ be a training dataset, where $x_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$ for binary classification. We denote the model parameters as θ and the prediction function as $f_\theta(x)$. The standard training objective minimizes the empirical risk:

$$\mathcal{L}_{std}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i) \tag{3}$$

where ℓ is the loss function (e.g., binary cross-entropy).

As motivated by the operational cost analysis in Section 2, we seek to augment this objective with penalty terms that discourage regression on previously learned examples, particularly when the operational benefits of accuracy improvements do not justify the costs of performance changes on specific subgroups or categories.

4.2 Classification Flip Penalty

Our first approach explicitly penalizes forgetting events—instances where an example transitions from being correctly classified to incorrectly classified between consecutive epochs.

Let $\theta^{(t)}$ denote the model parameters at epoch t, and define the correctness indicator:

$$c_i^{(t)} = \mathbb{I}[\text{sign}(f_{\theta^{(t)}}(x_i) - 0.5) = y_i] \tag{4}$$

The forgetting penalty at epoch t is:

$$\mathcal{R}_{flip}^{(t)}(\theta) = \sum_{i=1}^{n} \mathbb{I}[c_i^{(t-1)} = 1 \text{ and } c_i^{(t)} = 0] \tag{5}$$

The total training objective becomes:

$$\mathcal{L}_{flip}^{(t)}(\theta) = \mathcal{L}_{std}(\theta) + \lambda_{flip} \cdot \mathcal{R}_{flip}^{(t)}(\theta) \tag{6}$$

where $\lambda_{flip} \geq 0$ controls the strength of the forgetting penalty.

4.3 Confidence Drop Penalty

The Confidence Drop approach implements a more general "do no harm" principle by penalizing any increase in per-example loss, not just classification flips.

For each example i, we track the per-example loss from the previous epoch:

$$\ell_i^{(t-1)} = \ell(f_{\theta^{(t-1)}}(x_i), y_i) \tag{7}$$

The Confidence Drop penalty encourages the current loss to not exceed the previous loss:

$$\mathcal{R}_{conf}^{(t)}(\theta) = \sum_{i=1}^{n} \max(0, \ell(f_\theta(x_i), y_i) - \ell_i^{(t-1)}) \tag{8}$$

The complete objective is:

$$\mathcal{L}_{conf}^{(t)}(\theta) = \mathcal{L}_{std}(\theta) + \lambda_{conf} \cdot \mathcal{R}_{conf}^{(t)}(\theta) \tag{9}$$

4.4 Implementation Details

Both penalty methods are applied after a warmup period to allow the model to stabilize before enforcing regression constraints. Algorithm 1 presents the general training procedure.

Input: Dataset $\mathcal{D}$, penalty type, λ, warmup epochs T_{warmup}
Initialize model parameters $\theta^{(0)}$;
$t \leftarrow 0$;
while *not converged* **do**
  Compute standard loss: $\mathcal{L}_{std}(\theta^{(t)})$;
  if $t > T_{warmup}$ **then**
    Compute penalty: $\mathcal{R}^{(t)}(\theta^{(t)})$;
    Total loss: $\mathcal{L}_{total} = \mathcal{L}_{std} + \lambda \cdot \mathcal{R}^{(t)}$;
  end
  else
    Total loss: $\mathcal{L}_{total} = \mathcal{L}_{std}$;
  end
  Update parameters: $\theta^{(t+1)} \leftarrow \theta^{(t)} - \alpha \nabla_\theta \mathcal{L}_{total}$;
  $t \leftarrow t + 1$;
end

Algorithm 1: Penalty-Based Training with Regression Prevention

5 Experiments

5.1 Dataset and Setup

We evaluate our methods on the Adult income dataset (Kohavi, 1996), a binary classification task predicting whether an individual's income exceeds \$50K based on demographic features. The dataset contains 32,561 training examples and 16,281 test examples with 14 features after preprocessing.

We use a multi-layer perceptron (MLP) with two hidden layers of 64 units each, ReLU activations, and a sigmoid output layer. All models are trained using the Adam optimizer with a learning rate of 0.01 for 50 epochs. The penalty mechanisms are activated after 10 warmup epochs.

5.2 Evaluation Protocol

To measure forgetting in a production-realistic setting, we establish a fixed out-of-sample (OOS) evaluation set from the original validation data. Our evaluation protocol follows these steps:

1. Train a baseline "production" model on the training data
2. Record the baseline model's predictions on the fixed OOS evaluation set
3. Continue training the model with different penalty mechanisms
4. Measure forgetting as examples where the baseline model was correct but the updated model is incorrect on the same OOS set

This approach ensures that forgetting measurements are not confounded by changes in the evaluation data and reflects the practical concern of maintaining performance on a stable set of examples that the production system must handle correctly.

5.3 Evaluation Metrics

We track two primary metrics:

- **Total Forgetting**: Cumulative count of forgetting events on the fixed OOS set
- **Final Accuracy**: Classification accuracy on held-out test set after training completion

5.4 Results

Table 1 summarizes the performance of all three methods. Both penalty-based approaches achieve substantial reductions in forgetting compared to standard training.

Key observations:

Table 1. Experimental Results on Adult Income Dataset

Method	Total Forgetting	Final Train Acc	Final Val Acc
Baseline	566	0.794	0.788
Classification Flip Penalty	12	0.759	0.760
Confidence Drop Penalty	29	0.786	0.783

- Both penalized methods reduced forgetting by more than an order of magnitude compared to baseline
- The Classification Flip Penalty method achieved the lowest forgetting (12 events) but with a more significant accuracy reduction (2.8% validation accuracy drop)
- The Confidence Drop Penalty method provided an effective trade-off, reducing forgetting by 95% while maintaining validation accuracy within 0.5% of baseline
- Standard training achieved the highest accuracy but experienced 566 forgetting events on the fixed OOS evaluation set

6 Discussion

6.1 Practical Implications

Our results demonstrate that simple penalty mechanisms can effectively prevent regression in standard supervised learning settings. The Soft Pareto approach is particularly appealing for production systems, as it provides substantial forgetting reduction with minimal accuracy cost.

These methods are especially valuable in:

- Production-grade systems where regression on known-good cases is unacceptable
- Human-facing models where consistency matters for user trust
- High-stakes domains like medical diagnosis, fraud detection, or compliance monitoring
- Curriculum or staged learning setups where early learning should be preserved

6.2 Limitations and Future Work

While our penalty-based approaches show promise, several limitations merit discussion:

1. **Computational Overhead**: Tracking per-example losses or correctness requires additional memory and computation

2. **Hyperparameter Sensitivity**: The penalty strength λ requires tuning for each application

3. **Limited Evaluation**: Our experiments focus on a single dataset and architecture

Future work should explore these methods across diverse datasets, architectures, and learning scenarios, including online learning and continual learning settings.

7 Conclusion

We have introduced two penalty-based approaches for preventing regression in machine learning: margin-based and loss-based penalties that discourage performance degradation on previously learned examples. Motivated by concrete operational costs in production systems, our methods provide a lightweight mechanism for incorporating stability constraints into standard supervised learning.

Our experimental evaluation demonstrates that these approaches can substantially reduce regression events with modest accuracy costs. The loss-based penalty, in particular, offers an effective trade-off for production scenarios where consistency on known-good examples has economic value.

This work establishes a foundation for treating model regression as an economic optimization problem rather than merely a technical curiosity. As machine learning systems become increasingly embedded in business-critical processes, techniques for managing the operational externalities of model updates will become essential components of responsible AI deployment.

References

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Kohavi, R. (1996). Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid.

Lin, X., Zhen, H.-L., Li, Z., Zhang, Q.-F., and Kwong, S. (2019). Pareto multi-task learning. *Advances in Neural Information Processing Systems*, 32.

Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

Navon, A., Shamsian, A., Chechik, G., and Fetaya, E. (2021). Learning the pareto front with hypernetworks. *International Conference on Learning Representations*.

Paul, M., Ganguli, S., and Dziugaite, G. K. (2021). Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, 34:20596–20607.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Toneva, M., Sordoni, A., Tachet des Combes, R., Trischler, A., Bengio, Y., and Gordon, G. J. (2019). An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations (ICLR)*.

Yu, H. and Chen, J. (2022). Class-balanced loss based on example forgetting for long-tailed recognition.

Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. *International conference on machine learning*, pages 3987–3995.